Red Hat Ceph Storage 7

管理指南

管理 Red Hat Ceph Storage

Last Updated: 2024-06-14

Red Hat Ceph Storage 7 管理指南

管理 Red Hat Ceph Storage

法律通告

摘要

本文档论述了如何管理进程、监控集群状态、管理用户以及添加和删除 Red Hat Ceph Storage 的守护进程。 红帽致力于替换我们的代码、文档和 Web 属性中存在问题的语言。我们从这四个术语开始：master、slave、黑名单和白名单。由于此项工作十分艰巨，这些更改将在即将推出的几个发行版本中逐步实施。详情请查看 CTO Chris Wright 信息。

目录

第 1 章 CEPH 管理 5

第 2 章 了解 CEPH 的进程管理 6
2.1. CEPH 进程管理 6
2.2. 使用 SYSTEMCTL 命令启动、停止和重启所有 CEPH 守护进程 6
2.3. 启动、停止和重启所有 CEPH 服务 7
2.4. 查看在容器中运行的 CEPH 守护进程的日志文件 9
2.5. 关闭并重启 RED HAT CEPH STORAGE 集群 10

第 3 章 监控 CEPH 存储集群 18
3.1. CEPH 存储集群的高级别监控 18
3.2. CEPH 存储集群的低级别监控 41

第 4 章 为 CEPH 存储扩展集群 55
4.1. 存储集群的扩展模式 56

第 5 章 覆盖 CEPH 行为 72
5.1. 设置和取消设置 CEPH 覆盖选项 72
5.2. CEPH 覆盖用例 74

第 6 章 CEPH 用户管理 76
6.1. CEPH 用户管理背景 76
6.2. 管理 CEPH 用户 80

第 7 章 CEPH-VOLUME 工具 88
7.1. CEPH 卷 LVM 插件 88
7.2. 为什么 CEPH-VOLUME 替换 CEPH-DISK？ 89
7.3. 使用 CEPH-VOLUME准备 CEPH OSD 91
7.4. 使用 CEPH-VOLUME 列出设备 94
7.5. 使用 CEPH-VOLUME 激活 CEPH OSD 96
7.6. 使用 CEPH-VOLUME 停用 CEPH OSD 98
7.7. 使用 CEPH-VOLUME 创建 CEPH OSD 99
7.8. 迁移 BLUEFS 数据 100
7.9. 扩展 BLUEFS DB 设备 106
7.10. 将批处理模式与 CEPH-VOLUME搭配使用 112
7.11. 使用 CEPH-VOLUME进行数据 113

第 8 章 CEPH 性能基准 117
8.1. 性能基准 117
8.2. CEPH 性能基准 117
8.3. 基准测试 CEPH 块性能 121
8.4. CEPHFS 性能基准测试 122
8.5. CEPH 对象网关性能基准测试 124

第 9 章 CEPH 性能计数器 127
9.1. 访问 CEPH 性能计数器 127
9.2. 显示 CEPH 性能计数器 128
9.3. 转储 CEPH 性能计数器 130
9.4. 平均计数和总和 131
9.5. CEPH 监控指标 131
9.6. CEPH OSD 指标 136
9.7. CEPH 对象网关指标 145

第 10 章 MCLOCK OSD 调度程序 151

10.1. MCLOCK OSD 调度程序与 WPQ OSD 调度程序的比较 151
10.2. 输入和输出资源的分配 152
10.3. 影响 MCLOCK 操作队列的因素 154
10.4. MCLOCK 配置 156
10.5. MCLOCK 客户端 157
10.6. MCLOCK 配置集 157
10.7. CEPH OSD 容量确定 177

第 11 章 BLUESTORE 188
11.1. CEPH BLUESTORE 188
11.2. CEPH BLUESTORE 设备 189
11.3. CEPH BLUESTORE 缓存 190
11.4. CEPH BLUESTORE 的大小考虑 191
11.5. 使用 BLUESTORE_MIN_ALLOC_SIZE 参数调优 CEPH BLUESTORE 191
11.6. 使用 BLUESTORE 管理工具重新划分 ROCKSDB 数据库 194
11.7. BLUESTORE 碎片工具 202
11.8. CEPH BLUESTORE BLUEFS 207
11.9. 使用 CEPH-BLUSTORE-TOOL 213

第 12 章 CRIMSON （技术预览） 220
12.1. CRIMSON 概述 220
12.2. CRIMSON 和 CLASSIC CEPH OSD 架构之间的区别 222
12.3. CRIMSON 指标 224
12.4. CRIMSON 配置选项 225
12.5. 配置 CRIMSON 226
12.6. CRIMSON 配置参数 228
12.7. 分析 CRIMSON 235

第 13 章 CEPHADM 故障排除 239
13.1. 暂停或禁用 CEPHADM 239
13.2. 每个服务和每个守护进程事件 240
13.3. 检查 CEPHADM 日志 241
13.4. 收集日志文件 242
13.5. 收集 SYSTEMD 状态 244
13.6. 列出所有下载的容器镜像 244
13.7. 手动运行容器 245
13.8. CIDR 网络错误 247
13.9. 访问管理 SOCKET 247
13.10. 手动部署 MGR 守护进程 248

第 14 章 CEPHADM 操作 252
14.1. 监控 CEPHADM 日志消息 252
14.2. CEPH 守护进程日志 254
14.3. 数据位置 255
14.4. CEPHADM 自定义配置文件 256

第 15 章 CEPHADM 健康检查 258
15.1. CEPHADM 操作健康检查 258
15.2. CEPHADM 配置健康检查 259

第 16 章 使用 CEPHADM-ANSIBLE 模块管理红帽 CEPH 存储集群 263
16.1. CEPHADM-ANSIBLE 模块 263
16.2. CEPHADM-ANSIBLE 模块选项 263
16.3. 使用 CEPHADM_BOOTSTRAP 和 CEPHADM_REGISTRY_LOGIN 模块引导存储集群 267
16.4. 使用 CEPH_ORCH_HOST 模块添加或删除主机 271

16.5. 使用 CEPH_CONFIG 模块设置配置选项 279
16.6. 使用 CEPH_ORCH_APPLY 模块应用服务规格 282
16.7. 使用 CEPH_ORCH_DAEMON 模块管理 CEPH 守护进程状态 285

附录 A. MCLOCK 配置选项 **288**

第 1 章 CEPH 管理

Red Hat Ceph Storage 集群是所有 Ceph 部署的基础。部署 Red Hat Ceph Storage 集群后，相关的管理操作可以保持 Red Hat Ceph Storage 集群运行正常且处于最佳状态。

Red Hat Ceph Storage 管理指南可帮助存储管理员执行这些任务，例如：

- 如何检查我的 Red Hat Ceph Storage 集群的健康状态？
- 如何启动和停止 Red Hat Ceph Storage 集群服务？
- 如何从正在运行的 Red Hat Ceph Storage 集群中添加或删除 OSD？
- 如何为 Red Hat Ceph Storage 集群中存储的对象管理用户身份验证和访问控制？
- 我想了解如何在 Red Hat Ceph Storage 集群中使用覆盖。
- 我想监控 Red Hat Ceph Storage 集群的性能。

基本 Ceph 存储集群由两种类型的守护进程组成：

- Ceph Object Storage Device (OSD) 将数据存储在分配给 OSD 的 PG 内
- Ceph Monitor 维护集群映射的一个主（master）副本

生产系统具有三个或更多 Ceph 监控功能，以实现高可用性，一般最少 50 个 OSD 用于可接受的负载平衡、数据重新平衡和数据恢复。

第 2 章 了解 CEPH 的进程管理

作为存储管理员，您可以按照 Red Hat Ceph Storage 集群中的类型或实例来操作各种 Ceph 守护进程。通过操作这些守护进程，您可以根据需要启动、停止和重启所有 Ceph 服务。

2.1. CEPH 进程管理

在红帽 Ceph 存储中，所有流程管理都通过 Systemd 服务完成。在每次 **start**, **restart**, 和 **stop** Ceph 守护进程时，必须指定守护进程类型或守护进程实例。

其它资源

- 有关使用 **systemd** 的更多信息，请参阅使用 systemctl 管理系统服务。

2.2. 使用 SYSTEMCTL 命令启动、停止和重启所有 CEPH 守护进程

您可以作为您要停止 Ceph 守护进程的主机中的 root 用户启动、停止和重启所有 Ceph 守护进程。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。
- 具有对节点的 **root** 访问权限。

流程

1. 在您要启动、停止和重启守护进程的主机上，运行 systemctl 服务来获取服务的 *SERVICE_ID*。

   示例

   ```
   [root@host01 ~]# systemctl --type=service
   ceph-499829b4-832f-11eb-8d6d-001a4a000635@mon.host01.service
   ```

2. 启动所有 Ceph 守护进程：

   语法

   ```
   systemctl start SERVICE_ID
   ```

   示例

   ```
   [root@host01 ~]# systemctl start ceph-499829b4-832f-11eb-8d6d-
   001a4a000635@mon.host01.service
   ```

3. 停止所有 Ceph 守护进程：

   语法

   ```
   systemctl stop SERVICE_ID
   ```

   示例

   -

```
[root@host01 ~]# systemctl stop ceph-499829b4-832f-11eb-8d6d-
001a4a000635@mon.host01.service
```

4. 重启所有 Ceph 守护进程：

语法

```
systemctl restart SERVICE_ID
```

示例

```
[root@host01 ~]# systemctl restart ceph-499829b4-832f-11eb-8d6d-
001a4a000635@mon.host01.service
```

2.3. 启动、停止和重启所有 CEPH 服务

Ceph 服务是具有相同类型的 Ceph 守护进程的逻辑组，它们配置为在同一 Red Hat Ceph Storage 集群中运行。Ceph 中的编配层允许用户以集中的方式管理这些服务，从而可以轻松地执行影响到同一逻辑服务的所有 Ceph 守护进程的操作。每个主机中运行的 Ceph 守护进程通过 Systemd 服务进行管理。您可以从要管理 Ceph 服务的主机启动、停止和重新启动所有 Ceph 服务。

重要

如果要在特定主机中启动、停止或重启特定的 Ceph 守护进程，您需要使用 SystemD 服务。要获取在特定主机中运行的 SystemD 服务列表，连接到主机，并运行以下命令：

示例

```
[root@host01 ~]# systemctl list-units “ceph*”
```

输出将为您提供可用于管理每个 Ceph 守护进程的服务名称的列表。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。
- 具有对节点的 **root** 访问权限。

流程

1. 登录到 Cephadm shell：

示例

```
[root@host01 ~]# cephadm shell
```

2. 运行 **ceph orch ls** 命令，以获取 Red Hat Ceph Storage 集群中配置的 Ceph 服务列表，并获取特定的服务 ID。

示例

```
[ceph: root@host01 /]# ceph orch ls
```

```
NAME                   RUNNING  REFRESHED  AGE  PLACEMENT  IMAGE NAME
IMAGE ID
alertmanager              1/1  4m ago     4M   count:1    registry.redhat.io/openshift4/ose-
prometheus-alertmanager:v4.5   b7bae610cd46
crash                     3/3  4m ago     4M   *          registry.redhat.io/rhceph-alpha/rhceph-6-
rhel9:latest           c88a5d60f510
grafana                   1/1  4m ago     4M   count:1    registry.redhat.io/rhceph-alpha/rhceph-6-
dashboard-rhel9:latest  bd3d7748747b
mgr                       2/2  4m ago     4M   count:2    registry.redhat.io/rhceph-alpha/rhceph-6-
rhel9:latest           c88a5d60f510
mon                       2/2  4m ago     10w  count:2    registry.redhat.io/rhceph-alpha/rhceph-6-
rhel9:latest           c88a5d60f510
nfs.foo                   0/1  -          -    count:1    <unknown>
<unknown>
node-exporter             1/3  4m ago     4M   *          registry.redhat.io/openshift4/ose-
prometheus-node-exporter:v4.5  mix
osd.all-available-devices     5/5  4m ago     3M   *          registry.redhat.io/rhceph-
alpha/rhceph-6-rhel9:latest           c88a5d60f510
prometheus                1/1  4m ago     4M   count:1    registry.redhat.io/openshift4/ose-
prometheus:v4.6              bebb0ddef7f0
rgw.test_realm.test_zone      2/2  4m ago     3M   count:2    registry.redhat.io/rhceph-
alpha/rhceph-6-rhel9:latest           c88a5d60f510
```

3. 要启动特定的服务，请运行以下命令：

   语法

   ```
   ceph orch start SERVICE_ID
   ```

   示例

   ```
   [ceph: root@host01 /]# ceph orch start node-exporter
   ```

4. 要停止特定的服务，请运行以下命令：

   重要

   ceph orch stop SERVICE_ID 命令会导致 Red Hat Ceph Storage 集群无法访问，仅适用于 MON 和 MGR 服务。建议您使用 **systemctl stop SERVICE_ID** 命令来停止主机上的特定守护进程。

   语法

   ```
   ceph orch stop SERVICE_ID
   ```

   示例

   ```
   [ceph: root@host01 /]# ceph orch stop node-exporter
   ```

   在 **ceph 或ch stop node-exporter** 命令中，删除 **node exporter** 服务的所有守护进程。

5. 要重启特定的服务，请运行以下命令：

语法

```
ceph orch restart SERVICE_ID
```

示例

```
[ceph: root@host01 /]# ceph orch restart node-exporter
```

2.4. 查看在容器中运行的 CEPH 守护进程的日志文件

使用容器主机中的 **journald** 守护进程，从容器查看 Ceph 守护进程的日志文件。

先决条件

- 安装 Red Hat Ceph Storage 软件。
- 节点的根级别访问权限。

流程

1. 要查看整个 Ceph 日志文件，请以 **root** 身份运行一个 **journalctl** 命令，其格式如下：

   语法

   ```
   journalctl -u ceph SERVICE_ID
   ```

   示例

   ```
   [root@host01 ~]# journalctl -u ceph-499829b4-832f-11eb-8d6d-
   001a4a000635@osd.8.service
   ```

   在上例中，您可以查看 OSD 的整个日志，其 ID 为 **osd.8**。

2. 要只显示最新的日志条目，请使用 **-f** 选项。

   语法

   ```
   journalctl -fu SERVICE_ID
   ```

   示例

   ```
   [root@host01 ~]# journalctl -fu ceph-499829b4-832f-11eb-8d6d-
   001a4a000635@osd.8.service
   ```

注意

您还可以使用 **sosreport** 实用程序查看 **journald** 日志。有关 SOS 报告的详情，请查看 什么是 sosreport 以及如何在 Red Hat Enterprise Linux 中创建它？ 红帽客户门户网站上的解决方案。

其它资源

- **journalctl** 手册页。

2.5. 关闭并重启 RED HAT CEPH STORAGE 集群

您可以使用两种不同的方法关闭并重启 Red Hat Ceph Storage 集群： **systemctl** 命令和 Ceph Orchestrator。您可以选择关闭并重启集群的方法。

2.5.1. 使用 systemctl 命令关闭并重启集群

您可以使用 **systemctl** 命令来关闭并重启 Red Hat Ceph Storage 集群。此方法遵循 Linux 停止服务的方式。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。
- 根级别访问权限.

流程

关闭 Red Hat Ceph Storage 集群

1. 停止使用块设备镜像 RADOS 网关 - 此集群上的 Ceph 对象网关以及任何其他客户端的客户端。
2. 登录到 Cephadm shell：

   示例

   ```
   [root@host01 ~]# cephadm shell
   ```

3. 在继续操作前，集群必须处于健康状态（**Health_OK** 以及所有的 PG 为 **active+clean**）。使用客户端密钥环在主机上（如 Ceph 监控器或 OpenStack 控制器节点）运行 **ceph status**，以确保集群健康。

   示例

   ```
   [ceph: root@host01 /]# ceph -s
   ```

4. 如果使用 Ceph 文件系统 (**CephFS**)，请关闭 **CephFS** 集群：

   语法

   ```
   ceph fs set FS_NAME max_mds 1
   ceph fs fail FS_NAME
   ceph status
   ceph fs set FS_NAME joinable false
   ```

   示例

   ```
   [ceph: root@host01 /]# ceph fs set cephfs max_mds 1
   [ceph: root@host01 /]# ceph fs fail cephfs
   [ceph: root@host01 /]# ceph status
   [ceph: root@host01 /]# ceph fs set cephfs joinable false
   ```

5. 设置 **noout**、**norecover**、**norebalance**、**nobackfill**、**nodown** 和 **pause** 标志。在使用客户端密钥环的节点上运行以下命令，如 Ceph 监控器或 OpenStack 控制器节点：

   示例

   ```
   [ceph: root@host01 /]# ceph osd set noout
   [ceph: root@host01 /]# ceph osd set norecover
   [ceph: root@host01 /]# ceph osd set norebalance
   [ceph: root@host01 /]# ceph osd set nobackfill
   [ceph: root@host01 /]# ceph osd set nodown
   [ceph: root@host01 /]# ceph osd set pause
   ```

   重要

   上例仅用于停止服务和 OSD 节点上的每个 OSD，需要在每个 OSD 节点上重复。

6. 如果 MDS 和 Ceph 对象网关节点位于自己的专用节点上，请将其关闭。

7. 获取守护进程的 systemd 目标：

   示例

   ```
   [root@host01 ~]# systemctl list-units --type target | grep ceph
   ceph-0b007564-ec48-11ee-b736-525400fd02f8.target loaded active active Ceph cluster
   0b007564-ec48-11ee-b736-525400fd02f8
   ceph.target                                      loaded active active All Ceph clusters and services
   ```

8. 禁用包含集群 FSID 的目标：

   示例

   ```
   [root@host01 ~]# systemctl disable ceph-0b007564-ec48-11ee-b736-525400fd02f8.target

   Removed "/etc/systemd/system/multi-user.target.wants/ceph-0b007564-ec48-11ee-b736-
   525400fd02f8.target".
   Removed "/etc/systemd/system/ceph.target.wants/ceph-0b007564-ec48-11ee-b736-
   525400fd02f8.target".
   ```

9. 停止目标：

   示例

   ```
   [root@host01 ~]# systemctl stop ceph-0b007564-ec48-11ee-b736-525400fd02f8.target
   ```

   这会停止主机上需要停止的所有守护进程。

10. 关闭该节点：

    示例

    ```
    [root@host01 ~]# shutdown
    Shutdown scheduled for Wed 2024-03-27 11:47:19 EDT, use 'shutdown -c' to cancel.
    ```

11. 对集群中的所有节点重复上述步骤。

重新引导 Red Hat Ceph Storage 集群

1. 如果涉及网络设备，请确保在打开任何 Ceph 主机或节点之前将其开机和稳定。
2. 打开管理节点。
3. 启用 systemd 目标以获取所有守护进程运行：

   示例

   ```
   [root@host01 ~]# systemctl enable ceph-0b007564-ec48-11ee-b736-525400fd02f8.target
   Created symlink /etc/systemd/system/multi-user.target.wants/ceph-0b007564-ec48-11ee-b736-525400fd02f8.target → /etc/systemd/system/ceph-0b007564-ec48-11ee-b736-525400fd02f8.target.
   Created symlink /etc/systemd/system/ceph.target.wants/ceph-0b007564-ec48-11ee-b736-525400fd02f8.target → /etc/systemd/system/ceph-0b007564-ec48-11ee-b736-525400fd02f8.target.
   ```

4. 启动 systemd 目标：

   示例

   ```
   [root@host01 ~]# systemctl start ceph-0b007564-ec48-11ee-b736-525400fd02f8.target
   ```

5. 等待所有节点上线。验证所有服务都已启动，且节点之间没有网络连接的问题。
6. 取消设置 **noout**、**norecover**、**norebalance**、**nobackfill**、**nodown** 和 **pause** 标志。在使用客户端密钥环的节点上运行以下命令，如 Ceph 监控器或 OpenStack 控制器节点：

   示例

   ```
   [ceph: root@host01 /]# ceph osd unset noout
   [ceph: root@host01 /]# ceph osd unset norecover
   [ceph: root@host01 /]# ceph osd unset norebalance
   [ceph: root@host01 /]# ceph osd unset nobackfill
   [ceph: root@host01 /]# ceph osd unset nodown
   [ceph: root@host01 /]# ceph osd unset pause
   ```

7. 如果使用 Ceph 文件系统 (**CephFS**)，请通过将 **joinable** 标记设置为 **true** 来使 **CephFS** 集群重新启动：

   语法

   ```
   ceph fs set FS_NAME joinable true
   ```

   示例

   ```
   [ceph: root@host01 /]# ceph fs set cephfs joinable true
   ```

验证

- 验证集群处于健康状态（**Health_OK** 和所有 PG **active+clean**）。在具有客户端密钥环的节点上运行 **ceph 状态**，如 Ceph 监控器或 OpenStack 控制器节点，以确保集群健康。

示例

```
[ceph: root@host01 /]# ceph -s
```

其它资源

- 有关安装 Ceph 的更多信息，请参阅 *Red Hat Ceph Storage 安装指南*。

2.5.2. 使用 Ceph Orchestrator 关闭并重新引导集群

您还可以使用 Ceph Orchestrator 的功能来关闭并重启 Red Hat Ceph Storage 集群。在大多数情况下，它是一个单个系统登录，可以帮助关闭集群。

Ceph 编排器支持若干操作，如 **start**, **stop**, 和 **restart**。您可以将这些命令与 **systemctl** 搭配使用，在某些情况下可以关闭或重新引导集群。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。
- 节点的根级别访问权限。

流程

关闭 Red Hat Ceph Storage 集群

1. 停止使用该集群上的用户 Block Device Image 和 Ceph 对象网关以及任何其他客户端的客户端。
2. 登录到 Cephadm shell：

   示例

   ```
   [root@host01 ~]# cephadm shell
   ```

3. 在继续操作前，集群必须处于健康状态（**Health_OK** 以及所有的 PG 为 **active+clean**）。使用客户端密钥环在主机上（如 Ceph 监控器或 OpenStack 控制器节点）运行 **ceph status**，以确保集群健康。

   示例

   ```
   [ceph: root@host01 /]# ceph -s
   ```

4. 如果使用 Ceph 文件系统 (**CephFS**)，请关闭 **CephFS** 集群：

   语法

   ```
   ceph fs set FS_NAME max_mds 1
   ceph fs fail FS_NAME
   ceph status
   ceph fs set FS_NAME joinable false
   ceph mds fail FS_NAME:N
   ```

示例

```
[ceph: root@host01 /]# ceph fs set cephfs max_mds 1
[ceph: root@host01 /]# ceph fs fail cephfs
[ceph: root@host01 /]# ceph status
[ceph: root@host01 /]# ceph fs set cephfs joinable false
[ceph: root@host01 /]# ceph mds fail cephfs:1
```

5. 设置 **noout**、**norecover**、**norebalance**、**nobackfill**、**nodown** 和 **pause** 标志。在使用客户端密钥环的节点上运行以下命令，如 Ceph 监控器或 OpenStack 控制器节点：

示例

```
[ceph: root@host01 /]# ceph osd set noout
[ceph: root@host01 /]# ceph osd set norecover
[ceph: root@host01 /]# ceph osd set norebalance
[ceph: root@host01 /]# ceph osd set nobackfill
[ceph: root@host01 /]# ceph osd set nodown
[ceph: root@host01 /]# ceph osd set pause
```

6. 停止 MDS 服务。

   a. 获取 MDS 服务名称：

   示例

   ```
   [ceph: root@host01 /]# ceph orch ls --service-type mds
   ```

   b. 使用上一步中获取的名称停止 MDS 服务：

   语法

   ```
   ceph orch stop SERVICE-NAME
   ```

7. 停止 Ceph 对象网关服务。对部署的每个服务重复此操作。

   a. 获取 Ceph 对象网关服务名称：

   示例

   ```
   [ceph: root@host01 /]# ceph orch ls --service-type rgw
   ```

   b. 使用获取的名称停止 Ceph 对象网关服务：

   语法

   ```
   ceph orch stop SERVICE-NAME
   ```

8. 停止 Alertmanager 服务：

示例

```
[ceph: root@host01 /]# ceph orch stop alertmanager
```

9. 停止 node-exporter 服务，该服务是监控堆栈的一部分：

   示例

   ```
   [ceph: root@host01 /]# ceph orch stop node-exporter
   ```

10. 停止 Prometheus 服务：

    示例

    ```
    [ceph: root@host01 /]# ceph orch stop prometheus
    ```

11. 停止 Grafana 仪表板服务：

    示例

    ```
    [ceph: root@host01 /]# ceph orch stop grafana
    ```

12. 停止崩溃的服务：

    示例

    ```
    [ceph: root@host01 /]# ceph orch stop crash
    ```

13. 将 OSD 节点从 cephadm 节点关闭，每次关闭一个。对集群中的所有 OSD 重复此步骤。

    a. 获取 OSD ID：

       示例

       ```
       [ceph: root@host01 /]# ceph orch ps --daemon-type=osd
       ```

    b. 使用您获取的 OSD ID 关闭 OSD 节点：

       示例

       ```
       [ceph: root@host01 /]# ceph orch daemon stop osd.1
       Scheduled to stop osd.1 on host 'host02'
       ```

14. 逐一停止监视器。

    a. 识别托管该监控器的主机：

       示例

       ```
       [ceph: root@host01 /]# ceph orch ps --daemon-type mon
       ```

    b. 在每个主机上，停止该监控器。

       i. 确定 **systemctl** 单元名称：

          示例

```
[ceph: root@host01 /]# systemctl list-units ceph-* | grep mon
```

ii. 停止该服务：

语法

```
systemct stop SERVICE-NAME
```

15. 关闭所有主机。

重新引导 Red Hat Ceph Storage 集群

1. 如果涉及网络设备，请确保在打开任何 Ceph 主机或节点之前将其开机和稳定。
2. 打开所有 Ceph 主机。
3. 从 Cephadm shell 登录管理节点：

   示例

   ```
   [root@host01 ~]# cephadm shell
   ```

4. 验证所有服务都处于运行状态：

   示例

   ```
   [ceph: root@host01 /]# ceph orch ls
   ```

5. 确保集群健康状态为 'Health_OK'status：

   示例

   ```
   [ceph: root@host01 /]# ceph -s
   ```

6. 取消设置 **noout**、**norecover**、**norebalance**、**nobackfill**、**nodown** 和 **pause** 标志。在使用客户端密钥环的节点上运行以下命令，如 Ceph 监控器或 OpenStack 控制器节点：

   示例

   ```
   [ceph: root@host01 /]# ceph osd unset noout
   [ceph: root@host01 /]# ceph osd unset norecover
   [ceph: root@host01 /]# ceph osd unset norebalance
   [ceph: root@host01 /]# ceph osd unset nobackfill
   [ceph: root@host01 /]# ceph osd unset nodown
   [ceph: root@host01 /]# ceph osd unset pause
   ```

7. 如果使用 Ceph 文件系统 (**CephFS**)，请通过将 **joinable** 标记设置为 **true** 来使 **CephFS** 集群重新启动：

   语法

   ```
   ceph fs set FS_NAME joinable true
   ```

示例

```
[ceph: root@host01 /]# ceph fs set cephfs joinable true
```

验证

- 验证集群处于健康状态（**Health_OK** 和所有 PG **active+clean**）。在具有客户端密钥环的节点上运行 **ceph 状态**，如 Ceph 监控器或 OpenStack 控制器节点，以确保集群健康。

示例

```
[ceph: root@host01 /]# ceph -s
```

其它资源

- 有关安装 Ceph 的更多信息，请参阅 *Red Hat Ceph Storage 安装指南*

第 3 章 监控 CEPH 存储集群

作为存储管理员，您可以监控 Red Hat Ceph Storage 集群的整体健康状况，以及监控 Ceph 各个组件的健康状态。

运行 Red Hat Ceph Storage 集群后，您可以开始监控存储集群，以确保 Ceph 监控器和 Ceph OSD 守护进程在高级别中运行。Ceph 存储集群客户端连接到 Ceph Monitor 并接收最新版本的存储集群映射，然后才能将数据读取和写入到存储集群中的 Ceph 池。因此，在 Ceph 客户端可以读取和写入数据之前，监控器集群必须拥有集群状态的协议。

Ceph OSD 必须将主 OSD 上的放置组与次要 OSD 上的 PG 的副本进行对等（peer）。如果出现错误，则 peering 将处于 **active + clean** 以外的状态。

3.1. CEPH 存储集群的高级别监控

作为存储管理员，您可以监控 Ceph 守护进程的健康状况，以确保它们已启动并在运行。高级别监控还涉及检查存储集群容量，以确保存储集群不会超过其**全满比率（full ratio）**。Red Hat Ceph Storage 仪表板是进行高级别监控的最常见方法。但是，您也可以使用命令行界面、Ceph 管理 socket 或 Ceph API 来监控存储集群。

3.1.1. 检查存储集群健康状况

启动 Ceph 存储集群后，在开始读取或写入数据前，首先检查存储集群的健康状态。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。
- 节点的根级别访问权限。

流程

1. 登录到 Cephadm shell：

   示例

   ```
   root@host01 ~]# cephadm shell
   ```

2. 您可以使用以下命令来检查 Ceph 存储集群的健康状态：

   示例

   ```
   [ceph: root@host01 /]# ceph health
   HEALTH_OK
   ```

3. 您可以通过运行 **ceph status** 命令检查 Ceph 存储集群的状态：

   示例

   ```
   [ceph: root@host01 /]# ceph status
   ```

   输出提供以下信息：

- 集群 ID
- 集群健康状态
- monitor map epoch 和 monitor 仲裁的状态。
- OSD map epoch 和 OSD 状态。
- Ceph 管理器的状态。
- 对象网关的状态。
- 放置组映射版本。
- 放置组和池的数量。
- 存储的数据数量和所存储的对象数量。
- 存储的数据总量。
- IO 客户端操作。
- 如果集群正在升级，在升级过程中的更新。
  在启动 Ceph 集群时，您可能会遇到运行状况警告，如 **HEALTH_WARN XXX num placement groups stale**。等待几分钟，然后再次检查。当存储集群就绪时，**ceph health** 应返回一个如 **HEALTH_OK** 的消息。此时，可以开始使用群集。

3.1.2. 监视存储集群事件

您可以使用命令行界面观察 Ceph 存储集群发生的事件。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。
- 节点的根级别访问权限。

流程

1. 登录到 Cephadm shell：

   示例

   ```
   root@host01 ~]# cephadm shell
   ```

2. 要监控集群的持续事件，请运行以下命令：

   示例

   ```
   [ceph: root@host01 /]# ceph -w
     cluster:
       id:     8c9b0072-67ca-11eb-af06-001a4a0002a0
       health: HEALTH_OK

     services:
   ```

```
    mon: 2 daemons, quorum Ceph5-2,Ceph5-adm (age 3d)
    mgr: Ceph5-1.nqikfh(active, since 3w), standbys: Ceph5-adm.meckej
    osd: 5 osds: 5 up (since 2d), 5 in (since 8w)
    rgw: 2 daemons active (test_realm.test_zone.Ceph5-2.bfdwcn,
test_realm.test_zone.Ceph5-adm.acndrh)

  data:
    pools:   11 pools, 273 pgs
    objects: 459 objects, 32 KiB
    usage:   2.6 GiB used, 72 GiB / 75 GiB avail
    pgs:     273 active+clean

  io:
    client:   170 B/s rd, 730 KiB/s wr, 0 op/s rd, 729 op/s wr

2021-06-02 15:45:21.655871 osd.0 [INF] 17.71 deep-scrub ok
2021-06-02 15:45:47.880608 osd.1 [INF] 1.0 scrub ok
2021-06-02 15:45:48.865375 osd.1 [INF] 1.3 scrub ok
2021-06-02 15:45:50.866479 osd.1 [INF] 1.4 scrub ok
2021-06-02 15:45:01.345821 mon.0 [INF] pgmap v41339: 952 pgs: 952 active+clean; 17130
MB data, 115 GB used, 167 GB / 297 GB avail
2021-06-02 15:45:05.718640 mon.0 [INF] pgmap v41340: 952 pgs: 1
active+clean+scrubbing+deep, 951 active+clean; 17130 MB data, 115 GB used, 167 GB /
297 GB avail
2021-06-02 15:45:53.997726 osd.1 [INF] 1.5 scrub ok
2021-06-02 15:45:06.734270 mon.0 [INF] pgmap v41341: 952 pgs: 1
active+clean+scrubbing+deep, 951 active+clean; 17130 MB data, 115 GB used, 167 GB /
297 GB avail
2021-06-02 15:45:15.722456 mon.0 [INF] pgmap v41342: 952 pgs: 952 active+clean; 17130
MB data, 115 GB used, 167 GB / 297 GB avail
2021-06-02 15:46:06.836430 osd.0 [INF] 17.75 deep-scrub ok
2021-06-02 15:45:55.720929 mon.0 [INF] pgmap v41343: 952 pgs: 1
active+clean+scrubbing+deep, 951 active+clean; 17130 MB data, 115 GB used, 167 GB /
297 GB avail
```

3.1.3. Ceph 如何计算数据使用量

used 值反映了使用的*实际*原始存储量。**xxx GB / xxx GB** 代表可用的存储（其中较小的数字）和总存储容量。总容量反映了在复制、克隆或快照前存储数据的大小。因此，实际存储的数据量通常会超过名义上的存储量。这是因为 Ceph 会创建数据的副本，进行克隆和快照也需要使用存储。

3.1.4. 了解存储集群用量统计

要检查集群的数据使用量和数据分布在池间，请使用 **df** 选项。它类似于 Linux **df** 命令。

如果某些 OSD 标记为 **IN**，则 **ceph df** 和 **ceph status** 命令的输出中的 **SIZE/AVAIL/RAW USED** 会有所不同。**SIZE/AVAIL/RAW USED 从 SIZE （最大磁盘大小）、RAW USE （磁盘上使用的空间）和 AVAIL （所有处于 IN 状态的 OSD）计算。您可以在 ceph osd df tree 命令的输出中看到 SIZE/AVAIL/RAW USED 的总 OSD。**

示例

-

```
[ceph: root@host01 /]#ceph df
--- RAW STORAGE ---
CLASS   SIZE    AVAIL    USED  RAW USED  %RAW USED
hdd    5 TiB  2.9 TiB  2.1 TiB   2.1 TiB      42.98
TOTAL  5 TiB  2.9 TiB  2.1 TiB   2.1 TiB      42.98

--- POOLS ---
POOL                     ID  PGS   STORED  OBJECTS     USED  %USED  MAX AVAIL
.mgr                      1    1  5.3 MiB        3   16 MiB      0    629 GiB
.rgw.root                 2   32  1.3 KiB        4   48 KiB      0    629 GiB
default.rgw.log           3   32  3.6 KiB      209  408 KiB      0    629 GiB
default.rgw.control       4   32      0 B        8      0 B      0    629 GiB
default.rgw.meta          5   32  1.7 KiB       10   96 KiB      0    629 GiB
default.rgw.buckets.index    7   32  5.5 MiB       22   17 MiB      0    629 GiB
default.rgw.buckets.data     8   32  807 KiB        3  2.4 MiB      0    629 GiB
default.rgw.buckets.non-ec   9   32  1.0 MiB        1  3.1 MiB      0    629 GiB
source-ecpool-86          11   32  1.2 TiB  391.13k  2.1 TiB  53.49    1.1 TiB
```

ceph df detail 命令提供了更多关于其他池统计数据的详细信息，如配额对象、配额字节、压缩状态等。

输出的 RAW STORAGE 部分概述了存储集群用于存储数据的存储量。

- **CLASS: OSD 设备的类。**
- **SIZE： 由存储集群管理的存储容量量。**

  在上例中，如果 SIZE 是 90 GiB，它是不包括复制因子（默认为三）的总大小。带有复制因子的可用的总容量为 30 GiB（90 GiB/3）。根据全满比率（默认为 0.85%），最大可用空间为 30 GiB * 0.85 = 25.5 GiB

- **AVAIL： 存储集群中可用空间的数量。**

  在上例中，如果 SIZE 是 90 GiB，而 USED 空间为 6 GiB，则 AVAIL 空间为 84 GiB。带有复制因素的总可用空间（默认为 84 GiB/3 = 28 GiB）

- **USED：用户数据使用的原始存储量。**

在上例中，100 MiB 是在考虑了复制因子后的总可用空间。实际可用大小为 33 MiB。RAW USED ： 用户数据、内部开销或保留容量占用的原始存储量。

- % RAW USED： RAW USED 的百分比 .使用这个数值以及 full ratio 和 near full ratio，以确保您没有消耗倒所有的存储集群容量。

输出的 POOLS 部分提供了池列表以及每个池的使用情况。本节的输出不会反映副本、克隆或快照的情况。例如，如果您存储 1 MB 的数据的对象，名义的使用量为 1 MB，但实际使用量可能为 3 MB 或更多。具体的实际使用量取决于副本的数量（例如： size = 3）、克隆和快照。

- POOL：池的名称。
- ID: 池 ID。
- STORED: 用户存储在池中的实际数据量。这个值会根据(k+M)/K 值、对象副本数以及池统计计算时降级的对象数量更改。
- OBJECTS: 每个池存储的名义数量。它是 STORED 大小 * 复制因素。
- USED: 存储以 KB 为单位的数据数量，除非数字带有 M（megabyte）或 G（gigabytes）。
- %USED: 每个池使用的名义存储的百分比。
- MAX AVAIL: 可以写入这个池的数据数量的估计值。它是在第一个 OSD 变为满之前可以使用的数据量。它考虑了 CRUSH map 中跨磁盘的项目分布数据，并使用第一个 OSD 来填充作为目标。

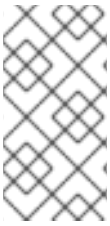
  在上例中，MAX AVAIL 为 153.85 MB（没有考虑复制因子，默认为三）。

  请参阅红帽知识库中的 *ceph df MAX AVAIL is incorrect for simple replicated pool* 以计算 MAX AVAIL 的值。

- QUOTA OBJECTS： 配额对象的数量。
- QUOTA BYTES： 配额对象中的字节数。
- USED COMPR： 为压缩数据分配的空间量，包括其压缩数据、分配、复制和擦除编码开销。
- UNDER COMPR: 通过压缩格式传输的数据量，以压缩形式存储有更多益处。

注意

POOLS 部分中的数字是估算的。它们不包括副本数、快照或克隆的数量。因此，USED 和 %USED 数值的总和可能会与输出的 GLOBAL 部分中的 RAW USED 和 %RAW USED 不同。



注意

MAX AVAIL 值是所用的复制或纠删代码的一个复杂功能，CRUSH 规则将存储映射到设备、这些设备的使用以及配置的 mon_osd_full_ratio。

其它资源

- 详情请参阅 Ceph 如何计算数据使用量。
- 详情请参阅 了解 OSD 用量统计。

3.1.5. 了解 OSD 使用统计

使用 ceph osd df 命令查看 OSD 使用率统计。

示例

```
[ceph: root@host01 /]# ceph osd df
ID CLASS WEIGHT  REWEIGHT SIZE    USE     DATA    OMAP    META    AVAIL   %USE VAR
```

```
PGS
 3   hdd 0.90959  1.00000  931GiB 70.1GiB 69.1GiB      0B    1GiB  861GiB 7.53 2.93  66
 4   hdd 0.90959  1.00000  931GiB 1.30GiB  308MiB      0B    1GiB  930GiB 0.14 0.05  59
 0   hdd 0.90959  1.00000  931GiB 18.1GiB 17.1GiB      0B    1GiB  913GiB 1.94 0.76  57
MIN/MAX VAR: 0.02/2.98  STDDEV: 2.91
```

- **ID: OSD 的名称。**
- **CLASS: OSD 使用的设备类型。**
- **WEIGHT: CRUSH 映射中的 OSD 权重。**
- **REWEIGHT： 默认的重新加权值。**
- **SIZE： OSD 的整体存储容量。**
- **USE： OSD 容量。**
- **DATA: 用户数据使用的 OSD 容量量。**
- **OMAP： 用于存储对象映射(omap)数据（rocksdb 中存储的键值对）的 bluefs 存储的估算值。**
- **META： 分配的 bluefs 空间或在 bluestore_bluefs_min 参数中设置的值（取决于哪个值更大），对于内部元数据，它的值是在 bluefs 中分配的总空间减去预计的 omap 数据大小。**
- **AVAIL： OSD 上可用的空间量。**
- **%USE： OSD 使用的存储百分比**

- VAR： 高于或低于平均利用率的差异。
- PGS： OSD 中的置放组数量。
- MIN/MAX VAR： 所有 OSD 的最小和最大变化。

其它资源

- 详情请参阅 *Ceph 如何计算数据使用量*。
- 详情请参阅 *了解 OSD* 用量统计。
- 详情请参阅 *Red Hat Ceph Storage 策略指南中的 CRUSH Weights*。

3.1.6. 检查存储集群状态

您可以从命令行界面查看 Red Hat Ceph Storage 集群的状态。status 子命令或 -s 参数将显示存储集群的当前状态。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。
- 节点的根级别访问权限。

流程

1. 登录到 Cephadm shell：

   示例

   ```
   [root@host01 ~]# cephadm shell
   ```

2. **要检查存储集群的状态，请执行以下操作：**

示例

```
[ceph: root@host01 /]# ceph status
```

或者

示例

```
[ceph: root@host01 /]# ceph -s
```

3. **在交互模式中，键入 ceph 并按 Enter 键：**

示例

```
[ceph: root@host01 /]# ceph
ceph> status
  cluster:
    id:     499829b4-832f-11eb-8d6d-001a4a000635
    health: HEALTH_WARN
            1 stray daemon(s) not managed by cephadm
            1/3 mons down, quorum host03,host02
            too many PGs per OSD (261 > max 250)

  services:
    mon:     3 daemons, quorum host03,host02 (age 3d), out of quorum: host01
    mgr:     host01.hdhzwn(active, since 9d), standbys: host05.eobuuv, host06.wquwpj
    osd:     12 osds: 11 up (since 2w), 11 in (since 5w)
```

```
    rgw:     2 daemons active (test_realm.test_zone.host04.hgbvnq,
test_realm.test_zone.host05.yqqilm)
    rgw-nfs: 1 daemon active (nfs.foo.host06-rgw)

  data:
    pools:   8 pools, 960 pgs
    objects: 414 objects, 1.0 MiB
    usage:   5.7 GiB used, 214 GiB / 220 GiB avail
    pgs:     960 active+clean

  io:
    client:   41 KiB/s rd, 0 B/s wr, 41 op/s rd, 27 op/s wr

ceph> health
HEALTH_WARN 1 stray daemon(s) not managed by cephadm; 1/3 mons down, quorum
host03,host02; too many PGs per OSD (261 > max 250)

ceph> mon stat
e3: 3 mons at {host01=[v2:10.74.255.0:3300/0,v1:10.74.255.0:6789/0],host02=
[v2:10.74.249.253:3300/0,v1:10.74.249.253:6789/0],host03=
[v2:10.74.251.164:3300/0,v1:10.74.251.164:6789/0]}, election epoch 6688, leader 1 host03,
quorum 1,2 host03,host02
```

3.1.7. 检查 Ceph Monitor 状态

如果存储集群有多个 Ceph Monitor（这是生产环境 Red Hat Ceph Storage 集群的要求），您可以在开始存储集群后检查 Ceph Monitor 仲裁状态，并在执行任何读取或写入数据前检查 Ceph Monitor 仲裁状态。

当运行了多个 Ceph monitor 时，必须存在仲裁。

定期检查 Ceph Monitor 状态，以确保它们正在运行。如果 Ceph Monitor 出现问题，这会防止达成存储集群状态协议，因此会阻止 Ceph 客户端读取和写入数据。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。
- 节点的根级别访问权限。

流程

1. **登录到 Cephadm shell：**

示例

```
[root@host01 ~]# cephadm shell
```

2. **要显示 Ceph Monitor 映射，请执行以下操作：**

示例

```
[ceph: root@host01 /]# ceph mon stat
```

或

示例

```
[ceph: root@host01 /]# ceph mon dump
```

3. **要检查存储集群的仲裁状态，请执行以下操作：**

```
[ceph: root@host01 /]# ceph quorum_status -f json-pretty
```

Ceph 返回仲裁状态。

示例

```
{
    "election_epoch": 6686,
    "quorum": [
        0,
        1,
        2
    ],
    "quorum_names": [
        "host01",
        "host03",
        "host02"
    ],
    "quorum_leader_name": "host01",
    "quorum_age": 424884,
    "features": {
        "quorum_con": "4540138297136906239",
        "quorum_mon": [
            "kraken",
            "luminous",
            "mimic",
            "osdmap-prune",
            "nautilus",
            "octopus",
            "pacific",
            "elector-pinging"
        ]
    },
    "monmap": {
        "epoch": 3,
        "fsid": "499829b4-832f-11eb-8d6d-001a4a000635",
        "modified": "2021-03-15T04:51:38.621737Z",
        "created": "2021-03-12T12:35:16.911339Z",
        "min_mon_release": 16,
        "min_mon_release_name": "pacific",
        "election_strategy": 1,
        "disallowed_leaders: ": "",
        "stretch_mode": false,
        "features": {
            "persistent": [
                "kraken",
                "luminous",
                "mimic",
                "osdmap-prune",
                "nautilus",
                "octopus",
                "pacific",
                "elector-pinging"
            ],
            "optional": []
        },
        "mons": [
            {
                "rank": 0,
```

```
        "name": "host01",
        "public_addrs": {
            "addrvec": [
                {
                    "type": "v2",
                    "addr": "10.74.255.0:3300",
                    "nonce": 0
                },
                {
                    "type": "v1",
                    "addr": "10.74.255.0:6789",
                    "nonce": 0
                }
            ]
        },
        "addr": "10.74.255.0:6789/0",
        "public_addr": "10.74.255.0:6789/0",
        "priority": 0,
        "weight": 0,
        "crush_location": "{}"
    },
    {
        "rank": 1,
        "name": "host03",
        "public_addrs": {
            "addrvec": [
                {
                    "type": "v2",
                    "addr": "10.74.251.164:3300",
                    "nonce": 0
                },
                {
                    "type": "v1",
                    "addr": "10.74.251.164:6789",
                    "nonce": 0
                }
            ]
        },
        "addr": "10.74.251.164:6789/0",
        "public_addr": "10.74.251.164:6789/0",
        "priority": 0,
        "weight": 0,
        "crush_location": "{}"
    },
    {
        "rank": 2,
        "name": "host02",
        "public_addrs": {
            "addrvec": [
                {
                    "type": "v2",
                    "addr": "10.74.249.253:3300",
                    "nonce": 0
                },
                {
                    "type": "v1",
```

```
                    "addr": "10.74.249.253:6789",
                    "nonce": 0
                }
            ]
        },
        "addr": "10.74.249.253:6789/0",
        "public_addr": "10.74.249.253:6789/0",
        "priority": 0,
        "weight": 0,
        "crush_location": "{}"
    }
  ]
 }
}
```


3.1.8. 使用 Ceph 管理 socket

使用管理套接字可以通过 UNIX 套接字文件直接与给定守护进程交互。例如，这个套接字可以：

- 在运行时列出 Ceph 配置
- 在运行时直接设置配置值，而不依赖 Monitor。当 Monitor 停机时，这非常有用。
- 转储历史操作
- 转储操作优先级队列状态
- 在不重启的情况下转储操作
- 转储性能计数器

此外，在对 Ceph monitor 或 OSD 相关的问题进行故障排除时，使用 socket 非常有用。

无论如何，如果守护进程没有运行，在尝试使用管理套接字时会返回以下错误：

```
Error 111: Connection Refused
```

重要

管理套接字仅在守护进程正在运行时才可用。当您正确关闭守护进程时，管理套接字会被删除。但是，如果守护进程意外终止，管理套接字可能仍然会被保留。

先决条件

- **一个正在运行的 Red Hat Ceph Storage 集群。**
- **节点的根级别访问权限。**

流程

1. **登录到 Cephadm shell：**

   示例

   ```
   [root@host01 ~]# cephadm shell
   ```

2. **使用套接字：**

   语法

   ```
   ceph daemon MONITOR_ID COMMAND
   ```

   替换：

- 守护进程的 ***MONITOR_ID***

- 带有要运行的命令的 ***COMMAND***。使用 **help** 列出给定守护进程的可用命令。

  查看 **Ceph Monitor** 的状态：

示例

```
[ceph: root@host01 /]# ceph daemon mon.host01 help
{
    "add_bootstrap_peer_hint": "add peer address as potential bootstrap peer for cluster
bringup",
    "add_bootstrap_peer_hintv": "add peer address vector as potential bootstrap peer for
cluster bringup",
    "compact": "cause compaction of monitor's leveldb/rocksdb storage",
    "config diff": "dump diff of current config and default config",
    "config diff get": "dump diff get <field>: dump diff of current and default config setting
<field>",
    "config get": "config get <field>: get the config value",
    "config help": "get config setting schema and descriptions",
    "config set": "config set <field> <val> [<val> ...]: set a config variable",
    "config show": "dump current config settings",
    "config unset": "config unset <field>: unset a config variable",
    "connection scores dump": "show the scores used in connectivity-based elections",
    "connection scores reset": "reset the scores used in connectivity-based elections",
    "counter dump": "dump all labeled and non-labeled counters and their values",
    "counter schema": "dump all labeled and non-labeled counters schemas",
    "dump_historic_ops": "show recent ops",
    "dump_historic_slow_ops": "show recent slow ops",
    "dump_mempools": "get mempool stats",
    "get_command_descriptions": "list available commands",
    "git_version": "get git sha1",
    "heap": "show heap usage info (available only if compiled with tcmalloc)",
    "help": "list available commands",
    "injectargs": "inject configuration arguments into running daemon",
    "log dump": "dump recent log entries to log file",
    "log flush": "flush log entries to log file",
    "log reopen": "reopen log file",
    "mon_status": "report status of monitors",
    "ops": "show the ops currently in flight",
    "perf dump": "dump non-labeled counters and their values",
    "perf histogram dump": "dump perf histogram values",
    "perf histogram schema": "dump perf histogram schema",
    "perf reset": "perf reset <name>: perf reset all or one perfcounter name",
    "perf schema": "dump non-labeled counters schemas",
    "quorum enter": "force monitor back into quorum",
    "quorum exit": "force monitor out of the quorum",
    "sessions": "list existing sessions",
```

```
    "smart": "Query health metrics for underlying device",
    "sync_force": "force sync of and clear monitor store",
    "version": "get ceph version"
}
```

示例

```
[ceph: root@host01 /]# ceph daemon mon.host01 mon_status

{
    "name": "host01",
    "rank": 0,
    "state": "leader",
    "election_epoch": 120,
    "quorum": [
        0,
        1,
        2
    ],
    "quorum_age": 206358,
    "features": {
        "required_con": "2449958747317026820",
        "required_mon": [
            "kraken",
            "luminous",
            "mimic",
            "osdmap-prune",
            "nautilus",
            "octopus",
            "pacific",
            "elector-pinging"
        ],
        "quorum_con": "4540138297136906239",
        "quorum_mon": [
            "kraken",
            "luminous",
            "mimic",
            "osdmap-prune",
            "nautilus",
            "octopus",
            "pacific",
            "elector-pinging"
        ]
    },
    "outside_quorum": [],
    "extra_probe_peers": [],
    "sync_provider": [],
    "monmap": {
        "epoch": 3,
```

```
"fsid": "81a4597a-b711-11eb-8cb8-001a4a000740",
"modified": "2021-05-18T05:50:17.782128Z",
"created": "2021-05-17T13:13:13.383313Z",
"min_mon_release": 16,
"min_mon_release_name": "pacific",
"election_strategy": 1,
"disallowed_leaders: ": "",
"stretch_mode": false,
"features": {
    "persistent": [
        "kraken",
        "luminous",
        "mimic",
        "osdmap-prune",
        "nautilus",
        "octopus",
        "pacific",
        "elector-pinging"
    ],
    "optional": []
},
"mons": [
    {
        "rank": 0,
        "name": "host01",
        "public_addrs": {
            "addrvec": [
                {
                    "type": "v2",
                    "addr": "10.74.249.41:3300",
                    "nonce": 0
                },
                {
                    "type": "v1",
                    "addr": "10.74.249.41:6789",
                    "nonce": 0
                }
            ]
        },
        "addr": "10.74.249.41:6789/0",
        "public_addr": "10.74.249.41:6789/0",
        "priority": 0,
        "weight": 0,
        "crush_location": "{}"
    },
    {
        "rank": 1,
        "name": "host02",
        "public_addrs": {
            "addrvec": [
                {
                    "type": "v2",
                    "addr": "10.74.249.55:3300",
                    "nonce": 0
                },
                {
```

```
                        "type": "v1",
                        "addr": "10.74.249.55:6789",
                        "nonce": 0
                    }
                ]
            },
            "addr": "10.74.249.55:6789/0",
            "public_addr": "10.74.249.55:6789/0",
            "priority": 0,
            "weight": 0,
            "crush_location": "{}"
        },
        {
            "rank": 2,
            "name": "host03",
            "public_addrs": {
                "addrvec": [
                    {
                        "type": "v2",
                        "addr": "10.74.249.49:3300",
                        "nonce": 0
                    },
                    {
                        "type": "v1",
                        "addr": "10.74.249.49:6789",
                        "nonce": 0
                    }
                ]
            },
            "addr": "10.74.249.49:6789/0",
            "public_addr": "10.74.249.49:6789/0",
            "priority": 0,
            "weight": 0,
            "crush_location": "{}"
        }
    ]
},
"feature_map": {
    "mon": [
        {
            "features": "0x3f01cfb9fffdffff",
            "release": "luminous",
            "num": 1
        }
    ],
    "osd": [
        {
            "features": "0x3f01cfb9fffdffff",
            "release": "luminous",
            "num": 3
        }
    ]
},
"stretch_mode": false
}
```

3. **或者，使用其套接字文件指定 Ceph 守护进程：**

语法

```
ceph daemon /var/run/ceph/SOCKET_FILE COMMAND
```

4. **查看特定主机上名为 osd.0 的 Ceph OSD 的状态：**

示例

```
[ceph: root@host01 /]# ceph daemon /var/run/ceph/ceph-osd.0.asok status
{
    "cluster_fsid": "9029b252-1668-11ee-9399-001a4a000429",
    "osd_fsid": "1de9b064-b7a5-4c54-9395-02ccda637d21",
    "whoami": 0,
    "state": "active",
    "oldest_map": 1,
    "newest_map": 58,
    "num_pgs": 33
}
```

注意

对于为特定守护进程可用的各种选项，您可以使用 help 而不是 status。

5. **列出 Ceph 进程的所有套接字文件：**

示例

```
[ceph: root@host01 /]# ls /var/run/ceph
```

其它资源

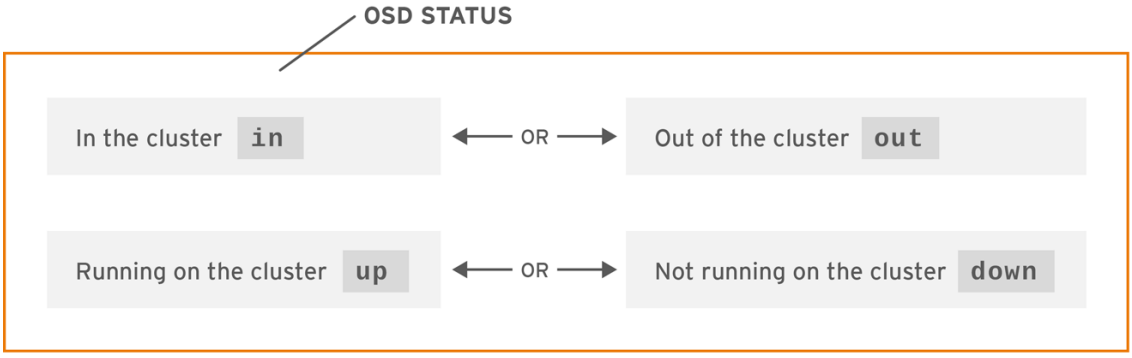
- 如需更多信息，请参阅 *Red Hat Ceph Storage 故障排除指南*。

3.1.9. 了解 Ceph OSD 状态

Ceph OSD 的状态可以是 in 存储集群，或 out 存储集群。它可以是 up 并运行，或 down 并没有运行。如果 Ceph OSD 为 up，它可以在存储集群内，可以在其中读取和写入数据，或者在存储集群外。如果它以前位于存储集群中，并最近从存储集群中移出，Ceph 将开始将放置组迁移到其他 Ceph OSD。如果 Ceph OSD 不在存储集群中，CRUSH 不会将放置组分配到 Ceph OSD。如果 Ceph OSD 是 down，它应该也是 out。

注意

如果 Ceph OSD 已关闭 且处于 in 状态，则存储集群将处于健康状态。



CEPH_459704_1017

如果执行诸如 ceph health, ceph -s 或 ceph -w 等命令，您可能会注意到存储集群并不总是回显 HEALTH OK。不需要紧张对于 Ceph OSD，您可以预计在一些预期情况下，存储集群不会 反映 HEALTH OK ：

- 还没有启动存储集群，且没有响应。
-

您刚启动或重启了存储集群，当还没有就绪，因为放置组正在被创建，Ceph OSD 正在进行对等处理。

- 您刚添加或删除 Ceph OSD。
- 您刚修改了存储集群映射。

监控 Ceph OSD 的一个重要方面是，当存储集群启动并正在运行，所有存在于存储集群中的所有 Ceph OSD 的状态为up 并在正常运行。

要查看所有 OSD 是否在运行，请执行：

示例

```
[ceph: root@host01 /]# ceph osd stat
```

或

示例

```
[ceph: root@host01 /]# ceph osd dump
```

结果应该显示 map epoch, eNNNN, OSD 的总数量, x, 多少个, y, 是 up, 多少个, z, 是 in:

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

如果存在于存储集群中的 Ceph OSD 数量超过了状态为 up 的数量。执行以下命令来标识没有运行的 ceph-osd 守护进程：

示例

```
[ceph: root@host01 /]# ceph osd tree

# id    weight  type name    up/down reweight
-1  3   pool default
-3  3       rack mainrack
-2  3           host osd-host
0   1               osd.0   up  1
1   1               osd.1   up  1
2   1               osd.2   up  1
```

提示

通过设计良好的 CRUSH 层次结构搜索功能可以帮助您更加快速地通过确定物理位置对存储集群进行故障排除。

如果 Ceph OSD 为 down，连接到该节点并启动它。您可以使用 Red Hat Storage Console 重启 Ceph OSD 守护进程，或者通过命令行。

语法

```
systemctl start CEPH_OSD_SERVICE_ID
```

示例

```
[root@host01 ~]# systemctl start ceph-499829b4-832f-11eb-8d6d-001a4a000635@osd.6.service
```

其它资源

- 如需了解更多详细信息，请参阅 *Red Hat Ceph Storage 仪表板指南*。

3.2. CEPH 存储集群的低级别监控

作为存储管理员，您可以从低级角度监控 Red Hat Ceph Storage 集群的健康状态。低级监控通常涉及确保 Ceph OSD 正确对等。当发生对等错误时，放置组将处于降级状态。这种降级状态可能是许多不同的事情，如硬件故障、挂起或崩溃的 Ceph 守护进程、网络延迟或完整的站点中断。

3.2.1. 监控放置组设置

当 CRUSH 将 PG 分配给 Ceph OSD 时，它会查看池的副本数量，并将 PG 分配到 Ceph OSD，使得 PG 的每个副本分配到不同的 Ceph OSD。例如，如果池需要三个放置组副本，则 CRUSH 可以分别分配给 osd.1、 osd.2 和 osd.3。CRUSH 实际上寻求伪随机放置，这会考虑您在 CRUSH 映射中设置的故障域，因此您很少会在大型集群中看到放置组被分配给最接近的邻居 Ceph OSD。对于包含特定放置组的 Ceph OSD 集合被称为活跃集（Acting Set）。在某些情况下，Acting Set 中的 OSD 为 down，否则无法在放置组中对对象进行服务请求。当出现这些情况时，不需要紧张。常见示例包括：

- 添加或删除 OSD。然后，CRUSH 将放置组重新分配到其他 Ceph OSD，从而改变了活跃集的组成，并使用 "backfill" 进程生成数据的迁移。
- Ceph OSD 过去为 down 并被重启，现在正在恢复。
- 在活跃集合中的一个 Ceph OSD 为 down，或无法服务请求时，另一个 Ceph OSD 会暂时担负其职责。

Ceph 使用 Up Set 处理客户端请求，这是实际处理请求的 Ceph OSD 集合。在大多数情况下，在线集和活跃集是相同的。如果它们不同，则表明 Ceph 正在迁移数据、Ceph OSD 正在恢复，或出现了问题，Ceph 通常会在此类情形中出现带有 "stuck stale" 消息的 HEALTH WARN 状态。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。
- 节点的根级别访问权限。

流程

1. **登录到 Cephadm shell：**

示例

```
[root@host01 ~]# cephadm shell
```

2. **检索放置组列表：**

示例

```
[ceph: root@host01 /]# ceph pg dump
```

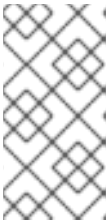
3. **查看对于一个给定的放置组，哪些 Ceph OSD 在 Acting Set 中或在 Up Set 中：**

语法

```
ceph pg map PG_NUM
```

示例

```
[ceph: root@host01 /]# ceph pg map 128
```



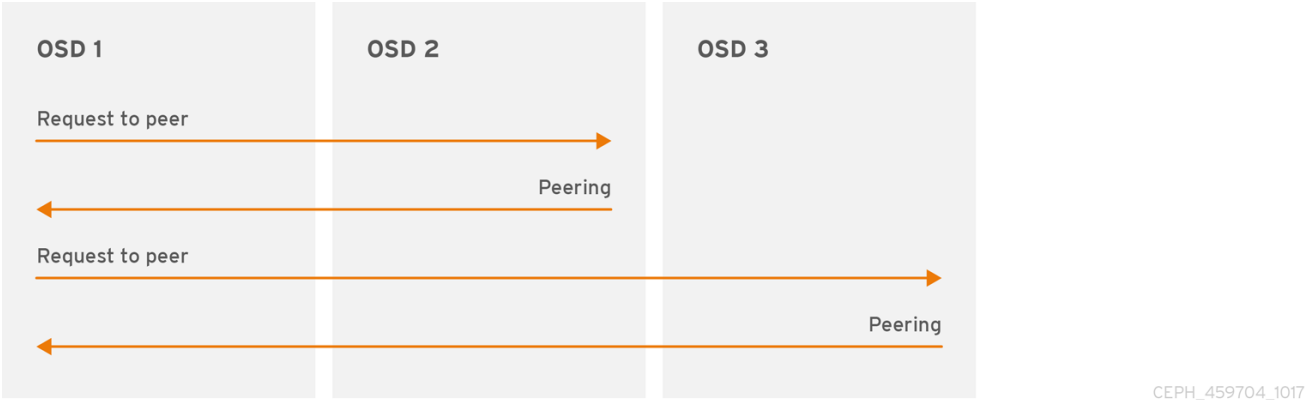
注意

如果设置与 Up Set 和 Acting 设置不匹配，这可能表明存储集群对自身或存储集群潜在的问题进行负载平衡。

3.2.2. Ceph OSD 对等

在将数据写入放置组之前，它必须处于 active 状态，且应该处于 clean 状态。若要让 Ceph 确定放置组的当前状态，PG 的 Primary OSD（活跃集中的第一个 OSD）与二级和三级 OSD 为对等，以变为放置组的当前状态建立协议。假设有三个 PG 副本的池。

图 3.1. 对等（peering）



3.2.3. 放置组状态

如果执行诸如 ceph health, ceph -s 或 ceph -w 等命令，您可能会注意到集群并不总是回显 HEALTH OK。检查 OSD 是否在运行后，您也应检查放置组状态。您应该可以预计，在与放置组对等相关的一些情况下，集群不会反映 HEALTH OK ：

- 您刚刚创建了一个池，放置组还没有对等。
- 放置组正在恢复。
- 您刚刚向集群添加一个 OSD 或从集群中移除一个 OSD。

- 您刚修改了 CRUSH map，并且已迁移了放置组。
- 在放置组的不同副本中，数据不一致。
- Ceph 清理放置组的副本。
- Ceph 没有足够存储容量来完成回填操作。

如果一个预期的情况导致 Ceph 反映了 HEALTH WARN，请不要紧张。在很多情况下，集群将自行恢复。在某些情况下，您可能需要采取措施。监控放置组的一个重要方面是确保在集群启动并运行所有放置组处于 active 状态，并且最好处于 clean 状态。

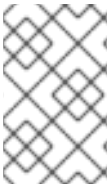
要查看所有放置组的状态，请执行：

示例

```
[ceph: root@host01 /]# ceph pg stat
```

结果显示放置组映射版本 vNNNNNN、放置组总数 x 以及放置组数量 y 都处于特定的状态，如 active+clean ：

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



注意

Ceph 通常会报告放置组的多个状态。

Snapshot Trimming PG States

当快照存在时，将报告两个额外的 PG 状态。

- **snaptrim ：PG 目前被修剪**
- **snaptrim_wait ：PG 等待被修剪**

输出示例：

```
244 active+clean+snaptrim_wait
 32 active+clean+snaptrim
```

除了放置组状态外，Ceph 还会回显所使用的数据量，aa(剩余存储容量)，bb（放置组的总存储容量）。在一些情况下，这些数字非常重要：

- **您达到了几乎全满比率或全满比率。**
- **由于 CRUSH 配置中的一个错误，您的数据不会分散到集群中。**

放置组 ID

放置组 ID 由池的号而不是名称组成，后跟一个句点 (.) 和放置组 ID（一个十六进制数字）。您可以从 ceph osd lspools 的输出中查看池编号及其名称。默认池名称为 data, metadata 和 rbd，分别与池号 0, 1 和 2 对应。完全限定的放置组 ID 的格式如下：

语法

```
POOL_NUM.PG_ID
```

输出示例：

```
0.1f
```

- **检索放置组列表：**

示例

```
[ceph: root@host01 /]# ceph pg dump
```

- **以 JSON 格式格式化输出并将其保存到文件中：**

语法

```
ceph pg dump -o FILE_NAME --format=json
```

示例

```
[ceph: root@host01 /]# ceph pg dump -o test --format=json
```

- **查询特定放置组：**

语法

```
ceph pg POOL_NUM.PG_ID query
```

示例

```
[ceph: root@host01 /]#  ceph pg 5.fe query
{
    "snap_trimq": "[]",
    "snap_trimq_len": 0,
    "state": "active+clean",
    "epoch": 2449,
    "up": [
        3,
        8,
        10
    ],
    "acting": [
        3,
        8,
        10
    ],
    "acting_recovery_backfill": [
        "3",
        "8",
        "10"
    ],
    "info": {
        "pgid": "5.ff",
        "last_update": "0'0",
        "last_complete": "0'0",
        "log_tail": "0'0",
        "last_user_version": 0,
        "last_backfill": "MAX",
        "purged_snaps": [],
        "history": {
            "epoch_created": 114,
            "epoch_pool_created": 82,
            "last_epoch_started": 2402,
            "last_interval_started": 2401,
            "last_epoch_clean": 2402,
            "last_interval_clean": 2401,
            "last_epoch_split": 114,
            "last_epoch_marked_full": 0,
            "same_up_since": 2401,
            "same_interval_since": 2401,
            "same_primary_since": 2086,
            "last_scrub": "0'0",
            "last_scrub_stamp": "2021-06-17T01:32:03.763988+0000",
            "last_deep_scrub": "0'0",
            "last_deep_scrub_stamp": "2021-06-17T01:32:03.763988+0000",
            "last_clean_scrub_stamp": "2021-06-17T01:32:03.763988+0000",
            "prior_readable_until_ub": 0
        },
        "stats": {
            "version": "0'0",
            "reported_seq": "2989",
            "reported_epoch": "2449",
            "state": "active+clean",
            "last_fresh": "2021-06-18T05:16:59.401080+0000",
```

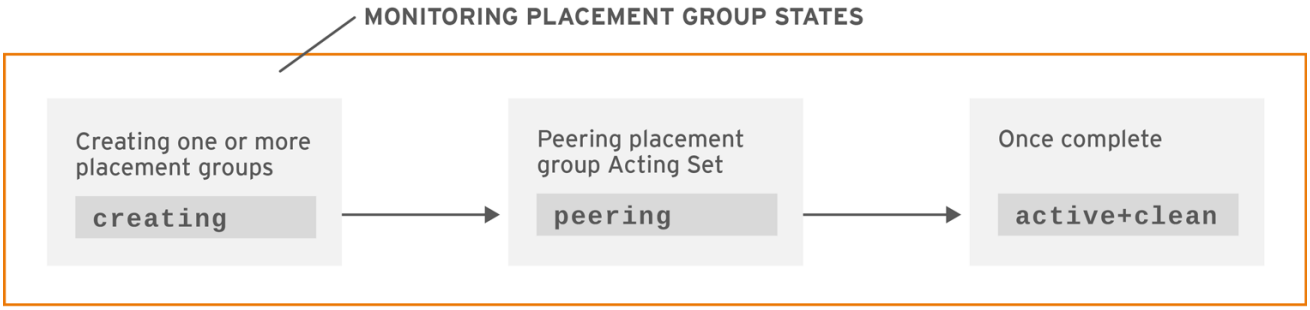
```
            "last_change": "2021-06-17T01:32:03.764162+0000",
            "last_active": "2021-06-18T05:16:59.401080+0000",
....
```

其它资源

- 有关快照修剪设置的更多详情，请参阅 ***Red Hat Ceph Storage Configuration Guide*** 中的 ***OSD Object storage daemon configuratopn options*** 部分的章节 ***Object Storage Daemon (OSD) configuration options***。

3.2.4. 放置组创建状态

在创建池时，将创建您指定的 PG 数量。Ceph 将在创建一个或多个放置组时回显 **creating**。创建之后，作为 PG Acting 设置一部分的 OSD 将会被对等点。对等点完成后，PG 状态应当为 **active+clean**，即 Ceph 客户端可以开始写入到 PG。



3.2.5. 放置组对等状态

当 Ceph 对等一个放置组时，Ceph 会将存储放置组副本的 OSD 变为与放置组中的对象和元数据的同意状态。当 Ceph 完成对等时，这意味着存储放置组的 OSD 同意该放置组的当前状态。但是，完成 peering 的进程并不表示每个副本都有最新的内容。

权威历史

Ceph 将不会确认对客户端的写操作，直到该工作集合的所有 OSD 都会保留这个写入操作。这种做法可确保，自上一次成功的对等操作，至少有一个成员具有每个确认的写操作的记录。

通过使用对每个已确认的写入操作的准确记录，Ceph 可以构建并分离 PG 的新权威历史记录。执行后，一组完全排序的操作（如果执行）会将使 OSD 的副本保持最新状态。

3.2.6. 放置组激活状态

Ceph 完成对等进程后，PG 可能会变为 active 状态。Active 状态表示放置组中的数据通常在主放置组中可用，而副本用于读取和写入操作。

3.2.7. 放置组清理状态

当放置组处于 clean 状态时，主 OSD 和副本 OSD 已成功对等，并且放置组没有预先复制。Ceph 复制 PG 中正确次数的所有对象。

3.2.8. 放置组降级状态

当客户端将对象写入 Primary OSD 时，OSD 负责将副本写入副本 OSD。在主 OSD 将对象写入存储后，PG 将会维持为 degraded 状态，直到主 OSD 收到来自副本 OSD 的 Ceph 已成功创建副本对象的确认。

PG 可以是 active+degraded 的原因是，OSD 可以处于 active 状态，尽管它还没有保存所有对象。如果 OSD 停机，Ceph 会将分配到 OSD 的每个 PG 标记为 degraded。当 Ceph OSD 重新上线时，Ceph OSD 必须再次进行对等。但是，如果客户端处于 active 状态，客户端仍然可以将新对象写入到一个降级的（degraded） PG。

如果 OSD 为 down 并且一直处于 degraded 状况，Ceph 可能会将 down OSD 标记为集群 out，并将 down OSD 的数据重新映射到另一个 OSD。在标记为 down 和标记为 out 之间的时间由 mon_osd_down_out_interval 控制。它被默认设置为 600 秒。

放置组也可以为 degraded，因为 Ceph 找不到一个或多个 Ceph 认为应该在放置组里的对象。虽然您无法读取或写入到未找到的对象，但您仍可以访问 degraded PG 中所有其他对象。

例如，如果在三方副本池中有九个 OSD：如果 OSD 数量 9 停机，分配给 OSD 9 的 PG 会进入降级状态。如果 OSD 9 没有恢复，则会退出存储集群和存储集群会重新平衡。在这种情况下，PG 被降级，然后恢复到 active 状态。

3.2.9. 放置组恢复状态

Ceph 设计为容错性，可以大规模地出现硬件和软件问题持续发展的问题。当 OSD 为 down 时，其内容可能落后于 PG 中其他副本的当前状态。当 OSD 变为 up，必须更新放置组的内容，以反映当前状态。在该时间段内，OSD 可能会处于 recovering 状态。

恢复并不总是不重要，因为硬件故障可能会导致多个 Ceph OSD 的级联故障。例如，一个机箱中的网络交换机可能会失败，这可能会导致多个主机机器的 OSD 落后于存储集群的当前状态。在错误解决后，每个 OSD 必须恢复。

Ceph 提供了多个设置，用于在新服务请求和恢复数据对象之间平衡资源争用，并将放置组恢复到当前状态。osd recovery delay start 设置允许 OSD 重新启动、重复操作，甚至在开始恢复过程前处理一些重播请求。osd recovery threads 设置限制恢复过程的线程数量，默认为一个线程。osd recovery thread timeout 设置一个线程超时，因为多个 Ceph OSD 失败，会以不同的频率重启和重新对等。osd recovery max active 设置限制了 Ceph OSD 可以同时处理的恢复请求数量，以防止 Ceph OSD 无法提供。osd recovery max chunk 设置限制了恢复的数据块的大小，以防止网络拥塞。

3.2.10. Back fill 状态

当新的 Ceph OSD 加入存储集群时，libvirt 会将放置组从集群中的 OSD 重新分配到新添加的 Ceph OSD。强制新 OSD 接受重新分配的 PG，可立即对新的 Ceph OSD 产生过量负载。使用放置组回填 OSD 使此过程在后台开始。回填完成后，新 OSD 会在请求就绪时开始提供请求。

在回填操作中，您可能会看到以下几个状态之一：

- backfill_wait 表示回填操作是待处理，但还没有进行。
- backfill 表示回填操作正在进行
- back_too_full 表示请求回填操作，但可能会因为存储容量不足而无法完成。

当无法回填放置组时，它被视为不完整。

Ceph 提供了多个设置，以管理与将放置组重新分配给 Ceph OSD（特别是新的 Ceph OSD）关联的负载激增。默认情况下，osd_max_backfills 将最大并发回填数量（来自或到一个 Ceph OSD）设置为 10。osd backfill full ratio 可让 Ceph OSD 在 OSD 接近其完整比率时拒绝回填请求，默认为 85%。如果 OSD 拒绝回填请求，osd backfill retry interval 可让 OSD 在 10 秒后重试请求。OSD 也可以设置 osd backfill scan min 和 osd backfill scan max，以管理扫描间隔（默认为 64 和 512）。

对于某些工作负载，完全避免常规恢复并使用回填会很有帮助。由于回填在后台发生，因此这允许 I/O 继续进行 OSD 中的对象。您可以通过将 osd_min_pg_log_entries 选项设置为 1，并将 osd_max_pg_log_entries 选项设置为 2 来强制进行回填，而不是恢复。当这个情况与您的工作负载相符，请联系您的红帽支持团队。

3.2.11. PG 重新映射状态

当决定服务设置放置组更改时，数据会从旧操作集迁移到新行为集。这可能需要经过一定时间后，新的 Primary OSD 才会处理服务请求。因此，可能需要旧的主系统继续服务请求，直到放置组迁移完成为止。数据迁移完成后，映射将使用新操作集合的 Primary OSD。

3.2.12. 放置组已过时状态

虽然 Ceph 使用心跳来确保主机和守护进程正在运行，ceph-osd 守护进程也会在没有及时报告统计数据的情况下变为 stuck 状态。例如，临时网络故障。默认情况下，OSD 守护进程每半秒钟报告其放置组、启动和失败统计，即 0.5，它比心跳阈值更频繁。如果放置组所采取集合的 Primary OSD 报告监控器失败，或者其他 OSD 报告了 Primary OSD down，则监视器会将 PG 标记为 stale。

当您启动存储集群时，通常会看到 stale 状态，直到对等进程完成为止。在存储集群运行一段时间后，如果放置组处于 stale 状态则代表这些放置组的主 OSD 的状态为 down 或者没有向监控器报告放置组统计信息。

3.2.13. 放置组错误替换状态

当 PG 临时映射到一个 OSD 时会有一些临时回填情况。当这个临时状态已不存在时，PG 可能仍然位于临时位置，而没有位于正确的位置。在这种情况下，它们被认为是 misplaced。这是因为实际上存在正确的额外副本数量，但一个或多个副本位于错误的地方。

例如，有 3 个 OSD：0、1,2 和所有 PG 映射到这三个。如果您添加了另一个 OSD(OSD 3)，一些 PG 现在将映射到 OSD 3，而不是另一个 OSD 3。但是，在 OSD 3 回填之前，PG 有一个临时映射，允许它继续从旧映射提供 I/O。在此期间，PG 为 misplaced，因为它有一个临时映射，但不是 degraded，因为存在 3 个副本。

示例

```
pg 1.5: up=acting: [0,1,2]
ADD_OSD_3
pg 1.5: up: [0,3,1] acting: [0,1,2]
```

[0,1,2] 是一个临时映射，因此 up 与 acting 并不完全相同。PG 为 misplaced 而不是 degraded，因为 [0,1,2] 仍然是三个副本。

示例

```
pg 1.5: up=acting: [0,3,1]
```

OSD 3 现在被回填，临时映射已被删除，而不是降级。

3.2.14. 放置组不完整状态

当内容不完整且对等失败（没有足够完整的 OSD 来执行恢复），PG 就会变为 incomplete 状态。

假设 OSD 1、2 和 3 是活跃的 OSD 集，它切换到 OSD 1、4 和 3，然后 osd.1 将请求一个临时活跃集包括 OSD 1、2 和 3，同时对 OSD 4 进行回填。在此期间，如果 OSD 1、2 和 3 都停机，则 osd.4 将是唯一没有完全回填所有数据的 OSD。此时，PG 会变为 incomplete，表明没有足够完整的 OSD 来执行恢复。

另外，如果没有涉及 osd.4，并且在 OSD 1、2 和 3 停机时，如果 osd.4 没有被涉及到，并且当 OSD 1、2 和 3 停机时，PG 就有可能变为 stale，这代表 mons 没有在这个 PG 上听到任何信息，因为执行集发生了变化。没有 OSD 以通知新的 OSD 的原因。

3.2.15. 找出卡住的 PG

仅仅因为放置组没有处于 active+clean 状态，并不一定代表它存在问题。通常，当 PG 卡住时，Ceph 无法进行自我修复。卡住状态包括：

- **Unclean: 放置组包含不会复制所需次数的对象。它们应该正在进行恢复。**
- **Inactive ：放置组无法处理读取或写入，因为它们正在等待具有最新数据的 OSD 返回到 up 状态。**
- **Stale：放置组处于未知状态，因为托管它们的 OSD 在一段时间内未报告到监控集群，并可使用 mon osd report timeout 配置。**

先决条件

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。
- 节点的根级别访问权限。

流程

1. 要识别卡的放置组，请执行以下操作：

语法

```
ceph pg dump_stuck {inactive|unclean|stale|undersized|degraded
[inactive|unclean|stale|undersized|degraded...]} {<int>}
```

示例

```
[ceph: root@host01 /]# ceph pg dump_stuck stale
OK
```

3.2.16. 查找对象的位置

Ceph 客户端检索最新的集群映射，并且 CRUSH 算法计算如何将对象映射到放置组，然后计算如何动态将 PG 分配给 OSD。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。
- 节点的根级别访问权限。

流程

1. **要查找对象位置，您需要对象名称和池名称：**

语法

```
ceph osd map POOL_NAME OBJECT_NAME
```

示例

```
[ceph: root@host01 /]# ceph osd map mypool myobject
```

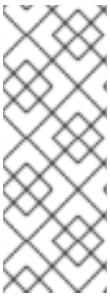
第 4 章 为 CEPH 存储扩展集群

作为存储管理员，您可以使用 2 个站点集群输入扩展模式来配置扩展集群。

Red Hat Ceph Storage 能够因为网络和集群而丢失 Ceph OSD，这些 OSD 在 CRUSH map 中随机分布失败。如果有多个 OSD 关闭，剩余的 OSD 和 monitor 仍然管理才能操作。

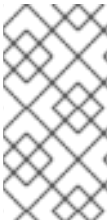
但是，这可能不是一些扩展集群配置的最佳解决方案，其中 Ceph 集群的大量部分只能使用单个网络组件。这个示例是位于多个数据中心的单个集群，用户希望继续丢失完整的数据中心。

标准配置有两个数据中心。其他配置位于云或可用区中。每个站点都包含两个数据副本，复制大小为 4。第三个站点应具有 tiebreaker 监控器，与主站点相比，这可能是虚拟机或高延迟。如果网络连接失败，且两个数据中心都保持活动状态，则此监控器选择一个站点来恢复数据。



注意

标准 Ceph 配置在网络或数据中心的很多故障中存活，永远不会破坏数据一致性。如果您在失败后恢复足够的 Ceph 服务器，它会恢复。如果您丢失数据中心，但仍然可以组成 monitor 的仲裁，并有足够的副本来满足池的 min_size 或 CRUSH 规则来再次复制以满足大小，Ceph 会保持可用性。



注意

没有额外的步骤可以关闭扩展集群。如需更多信息，请参阅 *关闭和重新引导 Red Hat Ceph Storage 集群*。

扩展集群故障

Red Hat Ceph Storage 永远不会破坏数据完整性和一致性。如果网络故障或节点丢失，并且服务仍然可以恢复，Ceph 会自行返回到正常功能。

但是，在有些情况下，即使您有足够的服务器来满足 Ceph 的一致性和大小限制，或者意外不满足约束，也会丢失数据可用性。

第一个重要故障类型是由网络不一致导致的。如果有网络分割，Ceph 可能无法将 OSD 标记为 down，以将其从操作放置组(PG)集中移除，尽管 Primary OSD 无法复制数据。发生这种情况时，不允许 I/O，

因为 Ceph 无法满足其持久性保证。

第二个重要的故障类别是，当您显示数据在不同数据间复制数据时，但约束不足以保证这一点。例如，您可能具有数据中心 A 和 B，CRUSH 规则以三个副本为目标，并将副本放在每个数据中心，min_size 为 2。PG 可能会活跃在站点 A 中有两个副本，而站点 B 中没有副本，这意味着如果丢失了站点 A，则丢失数据，Ceph 无法在其上操作。这种情形很难避免使用标准 CRUSH 规则。

4.1. 存储集群的扩展模式

要配置扩展集群，您必须进入扩展模式。启用扩展模式时，Ceph OSD 仅在数据中心的对等点或您指定的其他 CRUSH bucket 类型时将 PG 视为活跃状态，假设两者都处于活动状态。池将大小从默认的三个增加到四，每个站点有两个副本。

在扩展模式中，Ceph OSD 仅允许在同一数据中心内连接监控器。在没有指定位置的情况下，不允许新的 monitor 加入集群。

如果同时无法访问数据中心中的所有 OSD 和监控器，则存活的数据中心将进入 degraded 扩展模式。这会发出警告，将 min_size 减小到 1，并允许集群使用剩余的站点中的数据到达 活动状态。

注意

degraded 状态也会触发池太小的警告，因为池大小不会改变。但是，特殊的扩展模式标志可防止 OSD 在剩余的数据中心中创建额外的副本，因此仍然保留 2 个副本。

当缺少的数据中心再次变为可操作时，集群会进入 恢复 扩展模式。这会更改警告并允许 peering，但仍然需要来自数据中心的 OSD，而这始终为准。

当所有 PG 都处于已知状态且未降级或不完整时，集群会返回到常规扩展模式，结束警告，并将 min_size 恢复到其起始值 2。集群再次需要两个站点来对等，而不仅仅是整个站点，因此您可以切换到其他站点（如有必要）。

扩展模式限制

- 输入后，无法从扩展模式退出。
-

您不能在扩展模式下使用带有集群的纠删代码池。您不能使用纠删代码池进入扩展模式，也不会在扩展模式处于活动状态时创建纠删代码池。

- 支持扩展模式，且不支持两个以上的站点。

- 两个站点的权重应该相同。如果没有，您会收到以下错误：

示例

```
[ceph: root@host01 /]# ceph mon enable_stretch_mode host05 stretch_rule datacenter

Error EINVAL: the 2 datacenter instances in the cluster have differing weights 25947 and
15728 but stretch mode currently requires they be the same!
```

要在两个站点上实现相同的权重，部署在两个站点中的 Ceph OSD 的大小应该相等，即第一个站点中的存储容量等同于第二个站点的存储容量。

- 虽然它不强制执行，但您应该在每个站点上运行两个 Ceph 监视器，以及一个 tiebreaker，共 5 个。这是因为在扩展模式中，OSD 只能连接到自己的站点中的监控器。

- 您必须创建自己的 CRUSH 规则，在每个站点上提供两个副本，总计为在两个站点上共 4 个副本。

- 如果您有具有非默认大小或 min_size 的现有池，则无法启用扩展模式。

- 由于集群在降级时以 min_size 1 运行，因此您应该只使用 all-flash OSD 的扩展模式。这可最小化在连接恢复后恢复所需的时间，并尽可能减少数据丢失的可能性。

其它资源

- 有关故障排除步骤，请参阅 *扩展模式下* 对集群进行故障排除。

4.1.1. 为守护进程设置 CRUSH 位置

在进入扩展模式前，您需要将 CRUSH 位置设置为 Red Hat Ceph Storage 集群中的守护进程来准备集群。有两种方法可以做到这一点：

- 通过服务配置文件引导集群，该位置作为部署的一部分添加到主机中。
- 在部署集群后，通过 ceph osd crush add-bucket 和 ceph osd crush move 命令手动设置位置。

方法 1：引导集群

先决条件

- 对节点的根级别访问权限。

流程

1. 如果要引导新存储集群，您可以创建服务配置 .yaml 文件，该文件将节点添加到 Red Hat Ceph Storage 集群，并为服务应该运行的位置设置特定的标签：

示例

```
service_type: host
addr: host01
hostname: host01
location:
  root: default
  datacenter: DC1
labels:
  - osd
  - mon
  - mgr
---
service_type: host
addr: host02
hostname: host02
location:
  datacenter: DC1
labels:
  - osd
```

```
  - mon
---
service_type: host
addr: host03
hostname: host03
location:
  datacenter: DC1
labels:
  - osd
  - mds
  - rgw
---
service_type: host
addr: host04
hostname: host04
location:
  root: default
  datacenter: DC2
labels:
  - osd
  - mon
  - mgr
---
service_type: host
addr: host05
hostname: host05
location:
  datacenter: DC2
labels:
  - osd
  - mon
---
service_type: host
addr: host06
hostname: host06
location:
  datacenter: DC2
labels:
  - osd
  - mds
  - rgw
---
service_type: host
addr: host07
hostname: host07
labels:
  - mon
---
service_type: mon
placement:
  label: "mon"
---
service_id: cephfs
placement:
  label: "mds"
---
```

```
service_type: mgr
service_name: mgr
placement:
  label: "mgr"
---
service_type: osd
service_id: all-available-devices
service_name: osd.all-available-devices
placement:
  label: "osd"
spec:
  data_devices:
    all: true
---
service_type: rgw
service_id: objectgw
service_name: rgw.objectgw
placement:
  count: 2
  label: "rgw"
spec:
  rgw_frontend_port: 8080
```

2. **使用 --apply-spec 选项引导存储集群：**

语法

```
cephadm bootstrap --apply-spec CONFIGURATION_FILE_NAME --mon-ip
MONITOR_IP_ADDRESS --ssh-private-key PRIVATE_KEY --ssh-public-key PUBLIC_KEY --
registry-url REGISTRY_URL --registry-username USER_NAME --registry-password
PASSWORD
```

示例

```
[root@host01 ~]# cephadm bootstrap --apply-spec initial-config.yaml --mon-ip 10.10.128.68 --
ssh-private-key /home/ceph/.ssh/id_rsa --ssh-public-key /home/ceph/.ssh/id_rsa.pub --
registry-url registry.redhat.io --registry-username myuser1 --registry-password mypassword1
```

重要

您可以在 cephadm bootstrap 命令中使用不同的命令选项。但是，始终包含 --apply-spec 选项以使用服务配置文件并配置主机位置。

其它资源

- 如需有关 Ceph *bootstrap* 和不同 cephadm bootstrap 命令选项的更多信息，请参阅引导新存储集群。

方法 2：在部署后设置位置

先决条件

- 对节点的根级别访问权限。

流程

1. 添加两个 bucket，以计划将非tiebreaker 监视器的位置设置为 CRUSH map，并将 bucket 类型指定为 datacenter ：

语法

```
ceph osd crush add-bucket BUCKET_NAME BUCKET_TYPE
```

示例

```
[ceph: root@host01 /]# ceph osd crush add-bucket DC1 datacenter
[ceph: root@host01 /]# ceph osd crush add-bucket DC2 datacenter
```

2. **将存储桶移到 root=default 下：**

语法

```
ceph osd crush move BUCKET_NAME root=default
```

示例

```
[ceph: root@host01 /]# ceph osd crush move DC1 root=default
[ceph: root@host01 /]# ceph osd crush move DC2 root=default
```

3. **根据所需的 CRUSH 放置移动 OSD 主机：**

语法

```
ceph osd crush move HOST datacenter=DATACENTER
```

示例

```
[ceph: root@host01 /]# ceph osd crush move host01 datacenter=DC1
```

4.1.2. 进入扩展模式

新的扩展模式旨在处理两个站点。在 2 个站点集群中，组件可用性中断的风险较低。

先决条件

- 对节点的根级别访问权限。
- CRUSH 位置设置为主机。

流程

1. 设置每个 monitor 的位置，与 CRUSH map 匹配：

   语法

   ```
   ceph mon set_location HOST datacenter=DATACENTER
   ```

   示例

   ```
   [ceph: root@host01 /]# ceph mon set_location host01 datacenter=DC1
   [ceph: root@host01 /]# ceph mon set_location host02 datacenter=DC1
   [ceph: root@host01 /]# ceph mon set_location host04 datacenter=DC2
   [ceph: root@host01 /]# ceph mon set_location host05 datacenter=DC2
   [ceph: root@host01 /]# ceph mon set_location host07 datacenter=DC3
   ```

2. 生成 CRUSH 规则，将两个副本放在每个数据中心中：

   语法

```
ceph osd getcrushmap > COMPILED_CRUSHMAP_FILENAME
crushtool -d COMPILED_CRUSHMAP_FILENAME -o
DECOMPILED_CRUSHMAP_FILENAME
```

示例

```
[ceph: root@host01 /]# ceph osd getcrushmap > crush.map.bin
[ceph: root@host01 /]# crushtool -d crush.map.bin -o crush.map.txt
```

a. 编辑解译 **CRUSH** 映射文件，以添加新规则：

示例

```
rule stretch_rule {
        id 1 (1)
        type replicated
        min_size 1
        max_size 10
        step take DC1 (2)
        step chooseleaf firstn 2 type host
        step emit
        step take DC2 (3)
        step chooseleaf firstn 2 type host
        step emit
}
```

(1) 规则 **ID** 必须是唯一的。在这个示例中，只有一条 **id** 为 **0** 的规则，因此使用了 **id 1**，但您可能需要根据现有规则的数量使用不同的规则 **ID**。

(2) (3)

在本例中，有两个数据中心存储桶，名为 DC1 和 DC2。

注意

此规则使集群具有数据中心 DC1 的 read-affinity。因此，所有读取或写入都会通过位于 DC1 中的 Ceph OSD 进行。

如果这不是需要的，并且跨区平均分配读取或写入，CRUSH 规则如下：

示例

```
rule stretch_rule {
id 1
type replicated
min_size 1
max_size 10
step take default
step choose firstn 0 type datacenter
step chooseleaf firstn 2 type host
step emit
}
```

在此规则中，数据中心会被随机选择，并自动选择。

有关 firstn 和 indep 选项的更多信息，请参阅 *CRUSH 规则*。

3. 注入 CRUSH map，使规则可供集群使用：

语法

```
crushtool -c DECOMPILED_CRUSHMAP_FILENAME -o
COMPILED_CRUSHMAP_FILENAME
ceph osd setcrushmap -i COMPILED_CRUSHMAP_FILENAME
```

示例

```
[ceph: root@host01 /]# crushtool -c crush.map.txt -o crush2.map.bin
[ceph: root@host01 /]# ceph osd setcrushmap -i crush2.map.bin
```

4. **如果您没有在连接模式下运行监控器，请将选择策略设置为 connectivity ：**

示例

```
[ceph: root@host01 /]# ceph mon set election_strategy connectivity
```

5. **通过将 tiebreaker monitor 的位置设置为在数据中心间分割，进入扩展模式：**

语法

```
ceph mon set_location HOST datacenter=DATACENTER
ceph mon enable_stretch_mode HOST stretch_rule datacenter
```

示例

```
[ceph: root@host01 /]# ceph mon set_location host07 datacenter=DC3
[ceph: root@host01 /]# ceph mon enable_stretch_mode host07 stretch_rule datacenter
```

在本例中，monitor mon.host07 是 tiebreaker。

重要

tiebreaker monitor 的位置应该与之前设置非方法监视器的数据中心不同。在上例中，它是数据中心 DC3。

重要

不要将此数据中心添加到 CRUSH map，因为它在尝试进入扩展模式时会产生以下错误：

```
Error EINVAL: there are 3 datacenters in the cluster but stretch mode currently only works with 2!
```

注意

如果要编写自己的工具来部署 Ceph，您可以在引导监视器时使用新的 --set-crush-location 选项，而不是运行 ceph mon set_location 命令。这个选项只接受单个 bucket=location 对，如 ceph-mon --set-crush-location 'datacenter=DC1'，它必须与运行 enable_stretch_mode 命令时指定的 bucket 类型匹配。

6. 验证扩展模式是否已成功启用：

示例

```
[ceph: root@host01 /]# ceph osd dump

epoch 361
fsid 1234ab78-1234-11ed-b1b1-de456ef0a89d
created 2023-01-16T05:47:28.482717+0000
modified 2023-01-17T17:36:50.066183+0000
```

```
flags sortbitwise,recovery_deletes,purged_snapdirs,pglog_hardlimit
crush_version 31
full_ratio 0.95
backfillfull_ratio 0.92
nearfull_ratio 0.85
require_min_compat_client luminous
min_compat_client luminous
require_osd_release quincy
stretch_mode_enabled true
stretch_bucket_count 2
degraded_stretch_mode 0
recovering_stretch_mode 0
stretch_mode_bucket 8
```

extend_mode_enabled 应设为 **true**。您还可以查看扩展存储桶、扩展模式存储桶的数量，以及扩展模式是否降级或恢复。

7. 验证 **monitor** 是否在适当的位置：

示例

```
[ceph: root@host01 /]# ceph mon dump

epoch 19
fsid 1234ab78-1234-11ed-b1b1-de456ef0a89d
last_changed 2023-01-17T04:12:05.709475+0000
created 2023-01-16T05:47:25.631684+0000
min_mon_release 16 (pacific)
election_strategy: 3
stretch_mode_enabled 1
tiebreaker_mon host07
disallowed_leaders host07
0: [v2:132.224.169.63:3300/0,v1:132.224.169.63:6789/0] mon.host07; crush_location
{datacenter=DC3}
1: [v2:220.141.179.34:3300/0,v1:220.141.179.34:6789/0] mon.host04; crush_location
{datacenter=DC2}
2: [v2:40.90.220.224:3300/0,v1:40.90.220.224:6789/0] mon.host01; crush_location
{datacenter=DC1}
3: [v2:60.140.141.144:3300/0,v1:60.140.141.144:6789/0] mon.host02; crush_location
{datacenter=DC1}
4: [v2:186.184.61.92:3300/0,v1:186.184.61.92:6789/0] mon.host05; crush_location
{datacenter=DC2}
dumped monmap epoch 19
```

您还可以查看哪个 monitor 是 tiebreaker，以及监控选择策略。

其它资源

- 如需有关 *监控选择* 策略的更多信息，请参阅配置监控选择策略。

4.1.3. 以扩展模式添加 OSD 主机

您可以在扩展模式下添加 Ceph OSD。该流程与在未启用扩展模式的群集中添加 OSD 主机类似。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。
- 在集群中启用的扩展模式。
- 对节点的根级别访问权限。

流程

1. 列出可用的设备来部署 OSD：

   语法

   ```
   ceph orch device ls [--hostname=HOST_1 HOST_2] [--wide] [--refresh]
   ```

   示例

   -

```
[ceph: root@host01 /]# ceph orch device ls
```

2. **将 OSD 部署到特定的主机或所有可用设备上：**

- **从特定主机上的特定设备创建 OSD：**

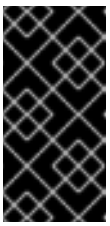
语法

```
ceph orch daemon add osd HOST:DEVICE_PATH
```

示例

```
[ceph: root@host01 /]# ceph orch daemon add osd host03:/dev/sdb
```

- **在任何可用和未使用的设备上部署 OSD：**



重要

这个命令会创建 **collocated WAL 和 DB** 设备。如果要创建非并置设备，请不要使用此命令。

示例

```
[ceph: root@host01 /]# ceph orch apply osd --all-available-devices
```

3. **将 OSD 主机移到 CRUSH 存储桶下：**

语法

```
ceph osd crush move HOST datacenter=DATACENTER
```

示例

```
[ceph: root@host01 /]# ceph osd crush move host03 datacenter=DC1
[ceph: root@host01 /]# ceph osd crush move host06 datacenter=DC2
```

注意

确保在两个站点中添加相同的拓扑节点。如果仅在一个站点中添加主机，则可能会出现问题。

其它资源

- **如需有关 *添加 Ceph OSD* 的更多信息，请参阅添加 OSD。**

第 5 章 覆盖 CEPH 行为

作为存储管理员，您需要了解如何对 Red Hat Ceph Storage 集群使用覆盖，以便在运行时更改 Ceph 选项。

5.1. 设置和取消设置 CEPH 覆盖选项

您可以通过设置和取消设置 Ceph 选项来覆盖 Ceph 的默认行为。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。
- 节点的根级别访问权限。

流程

1. 要覆盖 Ceph 的默认行为，请使用 ceph osd set 命令和您要覆盖的行为：

   语法

   ```
   ceph osd set FLAG
   ```

   设置行为后，ceph health 将反映您为集群设定的覆盖。

   示例

   ```
   [ceph: root@host01 /]# ceph osd set noout
   ```

2. **要覆盖 Ceph 的默认行为，请使用 ceph osd unset 命令以及您想要的覆盖。**

语法

```
ceph osd unset FLAG
```

示例

```
[ceph: root@host01 /]# ceph osd unset noout
```

标记	描述
noin	防止 OSD 被视为在集群 **in**。
noout	防止 OSD 被视在集群**外**。
noup	防止 OSD 被视为 **up** 并运行。
nodown	防止 OSD 被视为 **down**。
full	集群已达到其 **full_ratio**，从而阻止写操作。
pause	Ceph 将会停止处理读和写操作，但并不会影响 OSD **in**, **out**, **up** 或 **down** 状态。
nobackfill	Ceph 将阻止新的回填操作。
norebalance	Ceph 将阻止新的重新平衡操作。
norecover	Ceph 将阻止新的恢复操作。
noscrub	Ceph 将阻止新的清理操作。
nodeep-scrub	Ceph 将阻止新的深度清理操作。
notieragent	Ceph 将禁用正在查找 cold/dirty 对象以清空和驱除的进程。

标记	描述

5.2. CEPH 覆盖用例

- noin ：常见与 noout 一起使用来解决流化 OSD 的问题。
- noout ：如果超过了 mon osd report timeout，并且 OSD 没有报告给 monitor，OSD 将被标记为 out。如果发生这种情况错误，可以设置 noout 以防止在对问题进行故障排除时阻止 OSD 标记为 out。
- noup ：常见与 nodown 一起使用来解决流化 OSD 的问题。
- nodown: 网络问题可能会中断 Ceph 的 'heartbeat' 进程，而 OSD 可能会为 up，但仍标记为 down。您可以设置 nodown 来防止 OSD 在对问题进行故障排除时处于标记状态。
- full ：如果集群到达其 full_ratio，您可以预先将集群设置为 full 并扩展容量。

注意

将集群设置为 full 将阻止写操作。

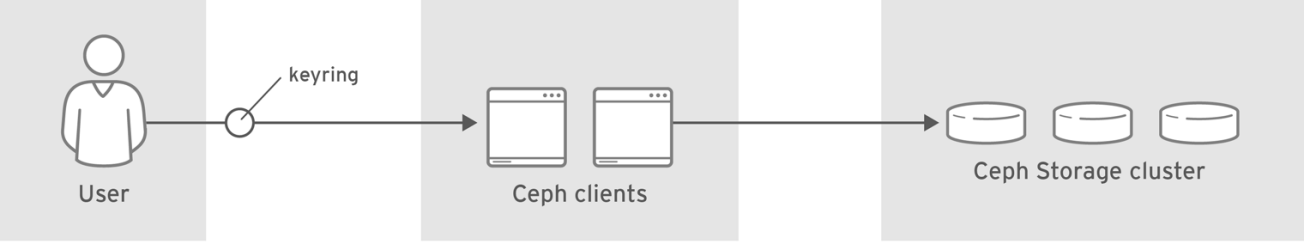
- pause ：如果需要在不读取和写入数据的情况下对正在运行的 Ceph 集群进行故障排除，您可以将集群设置为 pause 以防止客户端操作。
- nobackfill ：如果需要临时将 OSD 或节点设置为 down（如升级守护进程），您可以设置 nobackfill，以便 Ceph 在 OSD 为 down 时不会回填。
- norecover ：如果您需要替换 OSD 磁盘，并且在热交换磁盘时您不希望 PG 恢复到另一个 OSD，则可以设置 norecover。这可以防止其他 OSD 将新的 PG 复制到其他 OSD。
- noscrub 和 nodeep-scrubb：如果您希望防止刮除发生（例如，为了在高负载操作，如恢复、回填和重新平衡期间减少开销），您可以设置 noscrub 和/或 nodeep-scrub 以防止集群刮

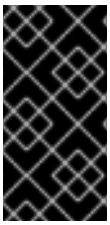
除 OSD。

- notieragent ：如果要阻止层代理进程查找冷对象，以刷新到后备存储层，则可以设置 notieragent。

第 6 章 CEPH 用户管理

作为存储管理员，您可以通过向 Red Hat Ceph Storage 集群中的对象提供身份验证和访问控制来管理 Ceph 用户。





重要

只要客户端在 Cephadm 范围内，Cephadm 便管理 Red Hat Ceph Storage 集群的客户端密钥环。用户不应修改 Cephadm 管理的密钥环，除非需要进行故障排除。

6.1. CEPH 用户管理背景

当 Ceph 在启用身份验证和授权的情况下运行时，您必须指定一个用户名。如果未指定用户名，Ceph 将使用 client.admin 管理用户作为默认用户名。

或者，您可以使用 CEPH_ARGS 环境变量以避免重新输入用户名和 secret。

无论 Ceph 客户端的类型（例如块设备、对象存储、文件系统、原生 API 或 Ceph 命令行），Ceph 都会将所有数据在池中作为对象保存。Ceph 用户必须有权访问池才能读取和写入数据。此外，管理 Ceph 用户必须具有执行 Ceph 管理命令的权限。

以下概念可帮助您理解 Ceph 用户管理。

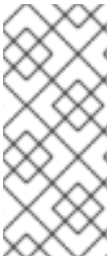
存储集群用户

Red Hat Ceph Storage 集群的用户是个人或一个应用程序。通过创建用户，您可以控制谁可以访问存储集群、池以及这些池中的数据。

Ceph 具有用户类型的概念。对于用户管理而言，类型将始终是 客户端。Ceph 使用带有句点(.)作为分隔符的名称来标识用户，它由用户类型和用户 ID 组成。例如，TYPE.ID、client.admin 或

client.user1。用户需要键入的原因是 Ceph 监控器和 OSD 也使用 Cephx 协议，但它们并不是客户端。区分用户类型有助于区分客户端用户和其他用户精简访问控制、用户监控和可追溯性。

有时，Ceph 的用户类型似乎比较混乱，因为 Ceph 命令行允许您根据命令行使用而指定具有或没有类型的用户。如果指定了 --user 或 --id，可以省略该类型。因此，输入 user1 可以代表 client.user1。如果指定 --name 或 -n，您必须指定类型和名称，如 client.user1。作为最佳做法，红帽建议尽可能使用类型和名称。



注意

Red Hat Ceph Storage 集群用户与 Ceph Object Gateway 用户不同。对象网关使用 Red Hat Ceph Storage 集群用户在网关守护进程和存储集群间进行通信，但网关具有自己的用户管理功能。

授权功能

Ceph 使用术语"capabilities（功能）"（大写）描述授权经过身份验证的用户来练习 Ceph 监控器和 OSD 的功能。功能也可以限制对池或池中命名空间中的数据的访问。Ceph 管理用户在创建或更新用户时设置用户的功能。功能语法遵循以下形式：

语法

```
DAEMON_TYPE 'allow CAPABILITY' [DAEMON_TYPE 'allow CAPABILITY']
```

- monitor Caps: Monitor 功能包括 r, w, x, allow profile *CAP*, 和 profile rbd。

  示例

  ```
  mon 'allow rwx`
  mon 'allow profile osd'
  ```

- OSD Caps: OSD 功能包括 r,w,x,class-read,class-write,profile osd,profile rbd, profile

rbd-read-only。另外，OSD 功能还允许池和命名空间设置。

语法

```
osd 'allow CAPABILITY' [pool=POOL_NAME] [namespace=NAMESPACE_NAME]
```

注意

Ceph 对象网关守护进程 (radosgw) 是 Ceph 存储集群的客户端，因此不表示 Ceph 存储集群守护进程类型。

以下条目描述了每个功能。

allow	守护进程的以前访问设置。
r	授予用户读取访问权限。需要 monitor 来检索 CRUSH map。
w	授予用户对对象的写入访问权限。
x	为用户提供调用类方法（即读写）的能力，并在 monitor 上执行 **auth** 操作。
class-read	授予用户调用类读取方法的能力。**x** 的子集.
class-write	授予用户调用类写入方法的能力。**x** 的子集.
*	授予用户对特定守护进程或池的读取、写入和执行权限，以及执行 admin 命令的能力。
配置集 osd	授予用户权限以 OSD 连接到其他 OSD 或 monitor。在 OSD 上延迟，使 OSD 能够处理复制心跳流量和状态报告。
配置集 bootstrap-osd	授予用户引导 OSD 的权限，以便在引导 OSD 时具有添加密钥的权限。
profile rbd	授予用户对 Ceph 块设备的读写访问权限。

profile rbd-read-only	授予用户对 Ceph 块设备的只读访问权限。

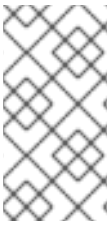
pool

池为 Ceph 客户端定义存储策略，并充当该策略的逻辑分区。

在 Ceph 部署中，创建池来支持不同类型的用例是很常见的。例如，云卷或镜像、对象存储、热存储、冷存储等等。将 Ceph 部署为 OpenStack 的后端时，典型的部署会具有卷、镜像、备份和虚拟机以及诸如 client.glance、client.cinder 等用户的池。

命名空间

池中的对象可以关联到池中命名空间的逻辑对象组。用户对池的访问可以关联到命名空间，以便用户读取和写入仅在命名空间内进行。写入到池中的一个命名空间的对象只能由有权访问该命名空间的用户访问。



注意

目前，命名空间仅适用于在 librados 之上编写的应用。块设备和对象存储等 Ceph 客户端目前不支持此功能。

命名空间的比率是池可以根据用例来计算聚合数据的计算方法，因为每个池创建了一组映射到 OSD 的放置组。如果多个池使用相同的 CRUSH 层次结构和规则集，OSD 性能可能会随着负载增加而降级。

例如，一个池对于每个 OSD 应有大约 100 个 PG。因此，有 1000 个 OSD 的集群对于一个池有 100,000 个 PG。映射到同一 CRUSH 层次结构的每个池将在 exemplary 集群中创建另一个 100,000 个放置组。相反，将对象写入命名空间只是将命名空间与对象名称关联的对象名称与单独池的计算开销相关联。您可以使用命名空间，而不是为用户或一组用户创建单独的池。

注意

目前仅在使用 librados 时才可用。

其它资源

- 有关配置身份验证的详细信息，请参阅 *Red Hat Ceph Storage 配置指南*。

6.2. 管理 CEPH 用户

作为存储管理员，您可以通过创建、修改、删除和导入用户来管理 Ceph 用户。Ceph 客户端用户可以是个人或应用程序，它们使用 Ceph 客户端与红帽 Ceph Storage 集群守护进程交互。

6.2.1. 列出 Ceph 用户

您可以使用命令行界面列出存储集群中的用户。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。
- 节点的根级别访问权限。

流程

1. 要列出存储集群中的用户，请执行以下操作：

示例

```
[ceph: root@host01 /]# ceph auth list
installed auth entries:

osd.10
 key: AQBW7U5gqOsEExAAg/CxSwZ/gSh8iOsDV3iQOA==
 caps: [mgr] allow profile osd
 caps: [mon] allow profile osd
 caps: [osd] allow *
osd.11
 key: AQBX7U5gtj/JIhAAPsLBNG+SfC2eMVEFkl3vfA==
 caps: [mgr] allow profile osd
 caps: [mon] allow profile osd
 caps: [osd] allow *
osd.9
 key: AQBV7U5g1XDULhAAKo2tw6ZhH1jki5aVui2v7g==
 caps: [mgr] allow profile osd
 caps: [mon] allow profile osd
```

```
 caps: [osd] allow *
client.admin
 key: AQADYEtgFfD3ExAAwH+C1qO7MSLE4TWRfD2g6g==
 caps: [mds] allow *
 caps: [mgr] allow *
 caps: [mon] allow *
 caps: [osd] allow *
client.bootstrap-mds
 key: AQAHYEtgpbkANBAANqoFlvzEXFwD8oB0w3TF4Q==
 caps: [mon] allow profile bootstrap-mds
client.bootstrap-mgr
 key: AQAHYEtg3dcANBAAVQf6brq3sxTSrCrPe0pKVQ==
 caps: [mon] allow profile bootstrap-mgr
client.bootstrap-osd
 key: AQAHYEtgD/QANBAATS9DuP3DbxEl86MTyKEmdw==
 caps: [mon] allow profile bootstrap-osd
client.bootstrap-rbd
 key: AQAHYEtgjxEBNBAANho25V9tWNNvIKnHknW59A==
 caps: [mon] allow profile bootstrap-rbd
client.bootstrap-rbd-mirror
 key: AQAHYEtgdE8BNBAAr6rLYxZci0b2hoIgH9GXYw==
 caps: [mon] allow profile bootstrap-rbd-mirror
client.bootstrap-rgw
 key: AQAHYEtgwGkBNBAAuRzI4WSrnowBhZxr2XtTFg==
 caps: [mon] allow profile bootstrap-rgw
client.crash.host04
 key: AQCQYEtgz8lGGhAAy5bJS8VH9fMdxuAZ3CqX5Q==
 caps: [mgr] profile crash
 caps: [mon] profile crash
client.crash.host02
 key: AQDuYUtgqgfdOhAAsyX+Mo35M+HFpURGad7nJA==
 caps: [mgr] profile crash
 caps: [mon] profile crash
client.crash.host03
 key: AQB98E5g5jHZAxAAklWSvmDsh2JaL5G7FvMrrA==
 caps: [mgr] profile crash
 caps: [mon] profile crash
client.nfs.foo.host03
 key: AQCgTk9gm+HvMxAAHbjG+XpdwL6prM/uMcdPdQ==
 caps: [mon] allow r
 caps: [osd] allow rw pool=nfs-ganesha namespace=foo
client.nfs.foo.host03-rgw
 key: AQCgTk9g8sJQNhAAPykcoYUuPc7IjubaFx09HQ==
 caps: [mon] allow r
 caps: [osd] allow rwx tag rgw *=*
client.rgw.test_realm.test_zone.host01.hgbvnq
 key: AQD5RE9gAQKdCRAAJzxDwD/dJObbInp9J95sXw==
 caps: [mgr] allow rw
 caps: [mon] allow *
 caps: [osd] allow rwx tag rgw *=*
client.rgw.test_realm.test_zone.host02.yqqilm
 key: AQD0RE9gkxA4ExAAFXp3pLJWdIhsyTe2ZR6IIw==
 caps: [mgr] allow rw
 caps: [mon] allow *
 caps: [osd] allow rwx tag rgw *=*
mgr.host01.hdhzwn
```

```
 key: AQAEYEtg3lhIBxAAmHodoIpdvnxK0llWF80ltQ==
 caps: [mds] allow *
 caps: [mon] profile mgr
 caps: [osd] allow *
mgr.host02.eobuuv
 key: AQAn6U5gzUuiABAA2Fed+jPM1xwb4XDYtrQxaQ==
 caps: [mds] allow *
 caps: [mon] profile mgr
 caps: [osd] allow *
mgr.host03.wquwpj
 key: AQAd6U5gIzWsLBAAbOKUKZlUcAVe9kBLfajMKw==
 caps: [mds] allow *
 caps: [mon] profile mgr
 caps: [osd] allow *
```

注意

用户使用 TYPE.ID 的形式代表，例如 osd.0 代表一个用户，类型为 osd，ID 是 0；client.admin 是一个用户，类型是 client，ID 为 admin，这是默认的 client.admin 用户。另请注意，每个条目都有一个 key: *VALUE* 条目，以及一个或多个 caps: 条目。

您可以将 -o *FILE_NAME* 选项与 ceph auth list 一起使用，以将输出保存到文件中。

6.2.2. 显示 Ceph 用户信息

您可以使用命令行界面显示 Ceph 的用户信息。

先决条件

- **一个正在运行的 Red Hat Ceph Storage 集群。**
- **节点的根级别访问权限。**

流程

1. **要检索特定用户、密钥和功能，请执行以下操作：**

语法

```
ceph auth export TYPE.ID
```

示例

```
[ceph: root@host01 /]# ceph auth export mgr.host02.eobuuv
```

2. **您也可以使用 -o *FILE_NAME* 选项。**

语法

```
ceph auth export TYPE.ID -o FILE_NAME
```

示例

```
[ceph: root@host01 /]# ceph auth export osd.9 -o filename
export auth(key=AQBV7U5g1XDULhAAKo2tw6ZhH1jki5aVui2v7g==)
```

auth export 命令与 auth get 相同，但也打印出内部 auid，但与最终用户无关。

6.2.3. 添加新的 Ceph 用户

添加用户会创建一个用户名，即 *TYPE.ID*、一个 secret key 和您用于创建用户的命令中包含的任何功

能。

用户密钥可让用户与 Ceph 存储集群进行身份验证。用户的能力授权用户在 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。

添加用户有几种方法：

- ceph auth add ：此命令是添加用户的规范方式。它将创建用户，并生成一个密钥并添加任何指定的功能。
- ceph auth get-or-create ：此命令通常是创建用户的最方便的方法，因为它会返回一个密钥文件，带有用户名（在括号中）和密钥。如果用户已存在，这个命令会以 keyfile 格式返回用户名和密钥。您可以使用 **-o *FILE_NAME*** 选项将输出保存到文件中。
- Ceph auth get-or-create-key ：这个命令是创建用户并仅返回用户密钥的便捷方式。这对只需要密钥的客户端（如 libvirt ）非常有用。如果用户已存在，这个命令只返回密钥。您可以使用 **-o *FILE_NAME*** 选项将输出保存到文件中。

在创建客户端用户时，您可以创建没有功能的用户。对于一个没有带有任何能力的用户，除进行身份验证之外没有任何用处，因为客户端无法从监控器检索 cluster map。但是，如果您想要以后使用 ceph auth caps 命令添加新的能力，则可以先创建一个没有权限的用户。

典型的用户在 Ceph OSD 上至少具有 Ceph 监视器的读取功能，以及 Ceph OSD 上的读写功能。另外，用户的 OSD 权限通常仅限于访问特定池。

```
[ceph: root@host01 /]# ceph auth add client.john mon 'allow r' osd 'allow rw pool=mypool'
[ceph: root@host01 /]# ceph auth get-or-create client.paul mon 'allow r' osd 'allow rw pool=mypool'
[ceph: root@host01 /]# ceph auth get-or-create client.george mon 'allow r' osd 'allow rw pool=mypool'
-o george.keyring
[ceph: root@host01 /]# ceph auth get-or-create-key client.ringo mon 'allow r' osd 'allow rw
pool=mypool' -o ringo.key
```

重要

如果您为用户提供 OSD 的功能，但没有限制特定池的访问权限，则该用户将能够访问集群中的所有池！

6.2.4. 修改 Ceph 用户

ceph auth caps 命令允许您指定用户并更改用户的能力。

先决条件

- **一个正在运行的 Red Hat Ceph Storage 集群。**
- 节**点的根级别访问权限。**

流程

1. **要添加能力，请使用表单：**

   语法

   ```
   ceph auth caps USERTYPE.USERID DAEMON 'allow [r|w|x|*|...] [pool=POOL_NAME]
   [namespace=NAMESPACE_NAME]'
   ```

   示例

   ```
   [ceph: root@host01 /]# ceph auth caps client.john mon 'allow r' osd 'allow rw pool=mypool'
   [ceph: root@host01 /]# ceph auth caps client.paul mon 'allow rw' osd 'allow rwx pool=mypool'
   [ceph: root@host01 /]# ceph auth caps client.brian-manager mon 'allow *' osd 'allow *'
   ```

2. **要删除功能，您可以重置能力。如果您希望用户无法访问之前设置的特定守护进程，请指定空字符串：**

   示例

   ```
   [ceph: root@host01 /]# ceph auth caps client.ringo mon ' ' osd ' '
   ```

-

其它资源

- 有关 *功能的更多详情，请参阅授权* 功能。

6.2.5. 删除 Ceph 用户

您可以使用命令行界面从 Ceph 存储集群中删除用户。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。
- 节点的根级别访问权限。

流程

1. 要删除用户，请使用 **ceph auth del**：

语法

```
ceph auth del TYPE.ID
```

示例

```
[ceph: root@host01 /]# ceph auth del osd.6
```

6.2.6. 输出 Ceph 用户密钥

您可以使用命令行界面显示 Ceph 用户的密钥信息。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。
- 节点的根级别访问权限。

流程

- 将用户的身份验证密钥输出到标准输出：

  语法

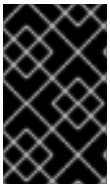
  ```
  ceph auth print-key TYPE.ID
  ```

  示例

  ```
  [ceph: root@host01 /]# ceph auth print-key osd.6

  AQBQ7U5gAry3JRAA3NoPrqBBThpFMcRL6Sr+5w==[ceph: root@host01 /]#
  ```

第 7 章 CEPH-VOLUME 工具

作为存储管理员，您可以使用 ceph-volume 实用程序准备、列出、创建、激活、停用、批处理、触发器、zap 和迁移 Ceph OSD。ceph-volume 实用程序是一个单一用途的命令行工具，用于将逻辑卷部署为 OSD。它使用插件类型框架来部署具有不同设备技术的 OSD。ceph-volume 实用程序遵循 ceph-disk 实用程序的类似工作流，用于部署 OSD，具有可预测的、可靠的准备、激活和启动 OSD 的方法。目前，ceph-volume 实用程序只支持 lvm 插件，计划以后支持其他技术。



重要

ceph-disk 命令已弃用。

7.1. CEPH 卷 LVM 插件

通过使用 LVM 标签，lvm 子命令可以通过查询与 OSD 关联的设备来存储和重新发现它们，以便可以激活它们。这包括对基于 lvm 的技术（如 dm-cache ）的支持。

使用 ceph-volume 时，dm-cache 的使用是透明的，并像逻辑卷一样对待 dm-cache。使用 dm-cache 时性能提升和丢失将取决于特定工作负载。通常，随机的读取和顺序的读取将以较小的块大小来提高性能。对于大的块大小，随机和顺序写入的性能会降低。

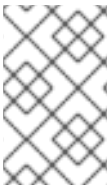
要使用 LVM 插件，请在 cephadm shell 中将 lvm 作为子命令添加到 ceph-volume 命令：

```
[ceph: root@host01 /]# ceph-volume lvm
```

以下是 lvm 子命令：

- prepare - 格式化 LVM 设备并将其与 OSD 关联。
- activate - 发现并挂载与 OSD ID 关联的 LVM 设备，并启动 Ceph OSD。
- list - 列出与 Ceph 关联的逻辑卷和设备。
- batch - 通过最少的交互为多 OSD 置备自动设置设备大小。

- deactivate - 取消激活 OSD。
- create - 从 LVM 设备创建新 OSD。
- trigger - 激活 OSD 的 systemd 帮助程序。
- zap - 从逻辑卷或分区中删除所有数据和文件系统。
- migrate - 将 BlueFS 数据迁移到另一个 LVM 设备。
- new-wal - 在指定逻辑卷中为 OSD 分配新的 WAL 卷。
- new-db - 在指定逻辑卷中为 OSD 分配新的 DB 卷。



注意

create 子命令将 prepare 和 activate 子命令合并到一个子命令中。

其它资源

- 如需了解更多详细信息，请参阅 create 子命令 部分。

7.2. 为什么 CEPH-VOLUME 替换 CEPH-DISK？

直到 Red Hat Ceph Storage 4，ceph-disk 实用程序用于准备、激活和创建 OSD。从 Red Hat Ceph Storage 4 开始，ceph-disk 被 ceph-volume 实用程序替代，该实用程序旨在作为 OSD 部署逻辑卷，同时在准备、激活和创建 OSD 时维护类似的 API 到 ceph-disk。

ceph-volume 的工作原理？

ceph-volume 是一个模块化工具，目前支持置备硬件设备、传统的 ceph-disk 设备和 LVM（逻辑卷管理器）设备的方法。ceph-volume lvm 命令使用 LVM 标签存储特定于 Ceph 的设备及其与 OSD 的关系信息。它使用这些标签来稍后重新发现和查询与 OSDs 关联的设备，以便它可以激活它们。它还支持基于 LVM 和 dm-cache 的技术。

ceph-volume 实用程序以透明方式使用 dm-cache，并将其视为逻辑卷。根据您要处理的特定工作负载，您可能会考虑使用 dm-cache 时的性能提升和丢失。通常，随机和顺序读取操作的性能会提高较小的块大小；而随机和顺序写入操作的性能则降低更大的块大小。使用 ceph-volume 不会导致任何明显的性能下降。

重要

ceph-disk 实用程序已弃用。

注意

如果这些设备仍在使用，ceph-volume simple 命令可处理旧的 ceph-disk 设备。

ceph-disk 的工作原理？

需要 ceph-disk 工具来支持许多不同类型的 init 系统，如 upstart 或 sysvinit，同时能够发现设备。因此，ceph-disk 只会专注于 GUID 分区表 (GPT) 分区。具体在 GPT GUID 中，以独特的方式标记设备，以如下方式回答问题：

- 此设备是否为 journal？
- 该设备是否是加密的数据分区？
- 设备是否部分准备了吗？

为解决这些问题，ceph-disk 使用 UDEV 规则与 GUID 匹配。

使用 ceph-disk 有什么缺点？

使用 UDEV 规则调用 ceph-disk 可能会导致在 ceph-disk systemd 单元和 ceph-disk 间的相互往来。整个过程非常不可靠且消耗时间，可能会导致 OSD 在节点启动过程中根本不会出现。此外，由于 UDEV 的异步行为，很难调试甚至复制这些问题。

由于 ceph-disk 只能用于 GPT 分区，所以它不支持其他技术，如逻辑卷管理器(LVM)卷或类似的设备映射器设备。

为确保 GPT 分区能与设备发现工作流正常工作，ceph-disk 需要大量特殊的标志。此外，这些分区需要设备完全归 Ceph 所有。

7.3. 使用 CEPH-VOLUME准备 CEPH OSD

prepare 子命令准备 OSD 后端对象存储，并消耗 OSD 数据和日志的逻辑卷(LV)。它不会修改逻辑卷，但使用 LVM 添加额外的元数据标签。这些标签使卷更容易发现，它们也会将卷识别为 Ceph Storage 集群的一部分，以及存储集群中这些卷的角色。

BlueStore OSD 后端支持以下配置：

- 块设备，block.wal 设备和 block.db 设备
- 块设备和一个 block.wal 设备
- 块设备和 block.db 设备
- 单个块设备

prepare 子命令接受整个设备或分区，或者用于 块 的逻辑卷。

先决条件

- 对 OSD 节点的 root 级别访问。
- （可选）创建逻辑卷。如果您提供到物理设备的路径，子命令会将设备转换为逻辑卷。这种方法更为简单，但是您无法配置或更改创建逻辑卷的方式。

流程

1. 提取 Ceph 密钥环：

   语法

```
ceph auth get client.ID -o ceph.client.ID.keyring
```

示例

```
[ceph: root@host01 /]# ceph auth get client.bootstrap-osd -o /var/lib/ceph/bootstrap-osd/ceph.keyring
```

2. **准备 LVM 卷：**

语法

```
ceph-volume lvm prepare --bluestore --data VOLUME_GROUP/LOGICAL_VOLUME
```

示例

```
[ceph: root@host01 /]# ceph-volume lvm prepare --bluestore --data example_vg/data_lv
```

a. **另外，如果您要将单独的设备用于 RocksDB，请指定 --block.db 和 --block.wal 选项：**

语法

```
ceph-volume lvm prepare --bluestore --block.db BLOCK_DB_DEVICE --block.wal
BLOCK_WAL_DEVICE --data DATA_DEVICE
```

示例

```
[ceph: root@host01 /]# ceph-volume lvm prepare --bluestore --block.db /dev/sda --
block.wal /dev/sdb --data /dev/sdc
```

b. 另**外，要加密数据，请使用 --dmcrypt 标志：**

语法

```
ceph-volume lvm prepare --bluestore --dmcrypt --data
VOLUME_GROUP/LOGICAL_VOLUME
```

示例

```
[ceph: root@host01 /]# ceph-volume lvm prepare --bluestore --dmcrypt --data
example_vg/data_lv
```

其它资源

- **如需更多详细信息，请参阅 *Red Hat Ceph Storage 管理指南*中的*使用 'ceph-volume' 激活 Ceph OSD*。**
-

如需更多详细信息，请参阅 *Red Hat Ceph Storage 管理指南*中的*使用 'ceph-volume' 创建 Ceph OSD*。

7.4. 使用 CEPH-VOLUME 列出设备

您可以使用 ceph-volume lvm list 子命令列出与 Ceph 集群关联的逻辑卷和设备，只要它们包含足够的元数据来允许该发现。输出会根据与设备关联的 OSD ID 进行分组。对于逻辑卷，devices key 会填充与逻辑卷关联的物理设备。

在某些情况下，ceph -s 命令的输出会显示以下出错信息：

```
1 devices have fault light turned on
```

在这种情况下，您可以使用 ceph device ls-lights 命令列出设备，该命令详细介绍了设备上的信息。根据信息，您可以关闭该设备的光盘。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。
- 对 Ceph OSD 节点的 root 级别访问权限。

流程

- 列出 Ceph 集群中的设备：

示例

```
[ceph: root@host01 /]# ceph-volume lvm list


====== osd.6 =======

  [block]       /dev/ceph-83909f70-95e9-4273-880e-5851612cbe53/osd-block-7ce687d9-07e7-4f8f-a34e-d1b0efb89920

      block device              /dev/ceph-83909f70-95e9-4273-880e-5851612cbe53/osd-block-7ce687d9-07e7-4f8f-a34e-d1b0efb89920
      block uuid                4d7gzX-Nzxp-UUG0-bNxQ-Jacr-l0mP-IPD8cX
```

```
    cephx lockbox secret
    cluster fsid              1ca9f6a8-d036-11ec-8263-fa163ee967ad
    cluster name              ceph
    crush device class        None
    encrypted                 0
    osd fsid                  7ce687d9-07e7-4f8f-a34e-d1b0efb89920
    osd id                    6
    osdspec affinity          all-available-devices
    type                      block
    vdo                       0
    devices                   /dev/vdc
```

- **可选：使用 lights 列出存储集群中的设备：**

示例

```
[ceph: root@host01 /]# ceph device ls-lights

{
    "fault": [
        "SEAGATE_ST12000NM002G_ZL2KTGCK0000C149"
    ],
    "ident": []
}
```

a. **可选：关闭该设备中的打印机：**

语法

```
ceph device light off DEVICE_NAME FAULT/INDENT --force
```

示例

```
[ceph: root@host01 /]# ceph device light off
SEAGATE_ST12000NM002G_ZL2KTGCK0000C149 fault --force
```

7.5. 使用 CEPH-VOLUME 激活 CEPH OSD

激活过程在引导时启用 systemd 单元，允许启用和挂载正确的 OSD 标识符及其 UUID。

先决条件

- **一个正在运行的 Red Hat Ceph Storage 集群。**
- **对 Ceph OSD 节点的 root 级别访问权限。**
- **Ceph OSD 由 ceph-volume 实用程序准备。**

流程

1. **从 OSD 节点获取 OSD ID 和 OSD FSID：**

```
[ceph: root@host01 /]# ceph-volume lvm list
```

2. **激活 OSD：**

语法

```
ceph-volume lvm activate --bluestore OSD_ID OSD_FSID
```

示例

```
[ceph: root@host01 /]# ceph-volume lvm activate --bluestore 10 7ce687d9-07e7-4f8f-a34e-d1b0efb89920
```

要激活为激活准备的所有 OSD，请使用 --all 选项：

示例

```
[ceph: root@host01 /]# ceph-volume lvm activate --all
```

3. **另外，您还可以使用 trigger 子命令。此命令不能直接使用，并由 systemd 使用，以便它将输入代理到 ceph-volume lvm activate。这会解析来自 systemd 和启动的元数据，并检测与 OSD 关联的 UUID 和 ID。**

语法

```
ceph-volume lvm trigger SYSTEMD_DATA
```

此处的 *SYSTEMD_DATA* 是 *OSD_ID-OSD_FSID* 格式。

示例

```
[ceph: root@host01 /]# ceph-volume lvm trigger 10 7ce687d9-07e7-4f8f-a34e-d1b0efb89920
```

其它资源

- 如需更多详细信息，请参阅 *Red Hat Ceph Storage 管理指南*中的*使用 'ceph-volume' 准备 Ceph OSD*。
- 如需更多详细信息，请参阅 *Red Hat Ceph Storage 管理指南*中的*使用 'ceph-volume' 创建 Ceph OSD*。

7.6. 使用 CEPH-VOLUME 停用 CEPH OSD

您可以使用 ceph-volume lvm 子命令停用 Ceph OSD。这个子命令会删除卷组和逻辑卷。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。
- 对 Ceph OSD 节点的 root 级别访问权限。
- Ceph OSD 通过 ceph-volume 实用程序激活。

流程

1. 从 OSD 节点获取 OSD ID：

```
[ceph: root@host01 /]# ceph-volume lvm list
```

2. 取消激活 OSD：

语法

```
ceph-volume lvm deactivate OSD_ID
```

示例

```
[ceph: root@host01 /]# ceph-volume lvm deactivate 16
```

其它资源

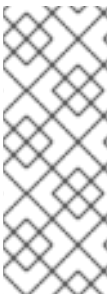
- 如需更多详细信息，请参阅 *Red Hat Ceph Storage 管理指南*中的*使用 'ceph-volume' 激活 Ceph OSD*。
- 如需更多详细信息，请参阅 *Red Hat Ceph Storage 管理指南*中的*使用 'ceph-volume' 准备 Ceph OSD*。
- 如需更多详细信息，请参阅 *Red Hat Ceph Storage 管理指南*中的*使用 'ceph-volume' 创建 Ceph OSD*。

7.7. 使用 CEPH-VOLUME 创建 CEPH OSD

create 子命令调用 prepare 子命令，然后调用 activate 子命令。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。
- 对 Ceph OSD 节点的 root 级别访问权限。



注意

如果您希望对创建过程拥有更多控制，可以单独使用 prepare 和 activate 子命令来创建 OSD，而不必使用 create。您可以使用两个子命令逐步将新的 OSD 引入到存储集群中，同时避免重新平衡大量数据。这两种方法的工作方式相同，唯一的不同是使用 create 子命令会使 OSD 在完成后立即变为 *up* 和 *in*。

流程

1. 要创建新 OSD，请执行以下操作：

语法

```
ceph-volume lvm create --bluestore --data VOLUME_GROUP/LOGICAL_VOLUME
```

示例

```
[root@osd ~]# ceph-volume lvm create --bluestore --data example_vg/data_lv
```

其它资源

- 如需更多详细信息，请参阅 *Red Hat Ceph Storage 管理指南*中的*使用 'ceph-volume' 准备 Ceph OSD*。
- 如需更多详细信息，请参阅 *Red Hat Ceph Storage 管理指南*中的*使用 'ceph-volume' 激活 Ceph OSD*。

7.8. 迁移 BLUEFS 数据

您可以使用 migrate LVM 子命令将位于 RocksDB 数据的 BlueStore 文件系统 (BlueFS) 数据迁移到目标卷。源卷（除主卷）被删除成功。

LVM 卷主要用于目标。

新卷附加到 OSD，替换其中一个源驱动器。

以下是 LVM 卷的放置规则：

- 如果源列表有 DB 或 WAL 卷，则目标设备会替换它。
- 如果源列表仅具有较慢的卷，则需要使用 new-db 或 new-wal 命令显式分配。

new-db 和 new-wal 命令分别将给定逻辑卷作为 DB 或 WAL 卷附加到给定的 OSD。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。
- 对 Ceph OSD 节点的 root 级别访问权限。
- Ceph OSD 由 ceph-volume 实用程序准备。
- 创建卷组和逻辑卷。

流程

1. 登录到 cephadm shell：

   示例

   ```
   [root@host01 ~]# cephadm shell
   ```

2. 停止您必须添加 DB 或 WAL 设备的 OSD：

   示例

```
[ceph: root@host01 /]# ceph orch daemon stop osd.1
```

3. **将新设备挂载到容器：**

示例

```
[root@host01 ~]# cephadm shell --mount /var/lib/ceph/72436d46-ca06-11ec-9809-ac1f6b5635ee/osd.1:/var/lib/ceph/osd/ceph-1
```

4. **将给定的逻辑卷作为 DB/WAL 设备附加到 OSD 中：**

注意

如果 OSD 附加了 DB，则此命令会失败。

语法

```
ceph-volume lvm new-db --osd-id OSD_ID --osd-fsid OSD_FSID --target VOLUME_GROUP_NAME/LOGICAL_VOLUME_NAME
```

示例

```
[ceph: root@host01 /]# ceph-volume lvm new-db --osd-id 1 --osd-fsid 7ce687d9-07e7-4f8f-a34e-d1b0efb89921 --target vgname/new_db
[ceph: root@host01 /]# ceph-volume lvm new-wal --osd-id 1 --osd-fsid 7ce687d9-07e7-4f8f-
```

```
a34e-d1b0efb89921 --target vgname/new_wal
```

5. **您可以使用以下方法迁移 BlueFS 数据：**

- **将 BlueFS 数据从主设备移动到已附加到 DB 的 LV：**

语法

```
ceph-volume lvm migrate --osd-id OSD_ID --osd-fsid OSD_UUID --from data --target
VOLUME_GROUP_NAME/LOGICAL_VOLUME_NAME
```

示例

```
[ceph: root@host01 /]# ceph-volume lvm migrate --osd-id 1 --osd-fsid 0263644D-0BF1-
4D6D-BC34-28BD98AE3BC8 --from data --target vgname/db
```

- **将 BlueFS 数据从共享主设备移动到 LV，应该作为新数据库附加：**

语法

```
ceph-volume lvm migrate --osd-id OSD_ID --osd-fsid OSD_UUID --from data --target
VOLUME_GROUP_NAME/LOGICAL_VOLUME_NAME
```

示例

```
[ceph: root@host01 /]# ceph-volume lvm migrate --osd-id 1 --osd-fsid 0263644D-0BF1-
4D6D-BC34-28BD98AE3BC8 --from data --target vgname/new_db
```

- **将 BlueFS 数据从 DB 设备移到新 LV，并替换 DB 设备：**

语法

```
ceph-volume lvm migrate --osd-id OSD_ID --osd-fsid OSD_UUID --from db --target
VOLUME_GROUP_NAME/LOGICAL_VOLUME_NAME
```

示例

```
[ceph: root@host01 /]# ceph-volume lvm migrate --osd-id 1 --osd-fsid 0263644D-0BF1-
4D6D-BC34-28BD98AE3BC8 --from db --target vgname/new_db
```

- **将 BlueFS 数据从主设备和 DB 设备移动到新 LV，并替换 DB 设备：**

语法

```
ceph-volume lvm migrate --osd-id OSD_ID --osd-fsid OSD_UUID --from data db --target
VOLUME_GROUP_NAME/LOGICAL_VOLUME_NAME
```

示例

```
[ceph: root@host01 /]# ceph-volume lvm migrate --osd-id 1 --osd-fsid 0263644D-0BF1-4D6D-BC34-28BD98AE3BC8 --from data db --target vgname/new_db
```

- **将 BlueFS 数据从 main、DB 和 WAL 设备移到新的 LV，删除 WAL 设备并替换 DB 设备：**

语法

```
ceph-volume lvm migrate --osd-id OSD_ID --osd-fsid OSD_UUID --from data db wal --target VOLUME_GROUP_NAME/LOGICAL_VOLUME_NAME
```

示例

```
[ceph: root@host01 /]# ceph-volume lvm migrate --osd-id 1 --osd-fsid 0263644D-0BF1-4D6D-BC34-28BD98AE3BC8 --from data db --target vgname/new_db
```

- **将 BlueFS 数据从 main、DB 和 WAL 设备移动到主设备中，删除 WAL 和 DB 设备：**

语法

```
ceph-volume lvm migrate --osd-id OSD_ID --osd-fsid OSD_UUID --from db wal --target VOLUME_GROUP_NAME/LOGICAL_VOLUME_NAME
```

示例

```
[ceph: root@host01 /]# ceph-volume lvm migrate --osd-id 1 --osd-fsid 0263644D-0BF1-4D6D-BC34-28BD98AE3BC8 --from db wal --target vgname/data
```

7.9. 扩展 BLUEFS DB 设备

您可以使用 ceph-bluestore 工具扩展作为 ceph-volume 创建的 ceph-volume 的 RocksDB 数据的 BlueStore 文件系统(BlueFS)数据的存储。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。
- Ceph OSD 由 ceph-volume 实用程序准备。
- 创建卷组和逻辑卷。

注意

在部署 OSD 的主机上运行这些步骤。

流程

1. 可选：在 cephadm shell 之外，列出 Red Hat Ceph Storage 集群中的设备。

示例

```
[ceph: root@host01 /]# ceph-volume lvm list

====== osd.3 =======

  [db]          /dev/db-test/db1
```

```
    block device              /dev/test/lv1
    block uuid                N5zoix-FePe-uExe-UngY-D9YG-BMs0-1tTDyB
    cephx lockbox secret
    cluster fsid              1a6112da-ed05-11ee-bacd-525400565cda
    cluster name              ceph
    crush device class
    db device                 /dev/db-test/db1
    db uuid                   1TUaDY-3mEt-fReP-cyB2-JyZ1-oUPa-hKPfo6
    encrypted                 0
    osd fsid                  94ff742c-7bfd-4fb5-8dc4-843d10ac6731
    osd id                    3
    osdspec affinity          None
    type                      db
    vdo                       0
    devices                   /dev/vdh

[block]       /dev/test/lv1

    block device              /dev/test/lv1
    block uuid                N5zoix-FePe-uExe-UngY-D9YG-BMs0-1tTDyB
    cephx lockbox secret
    cluster fsid              1a6112da-ed05-11ee-bacd-525400565cda
    cluster name              ceph
    crush device class
    db device                 /dev/db-test/db1
    db uuid                   1TUaDY-3mEt-fReP-cyB2-JyZ1-oUPa-hKPfo6
    encrypted                 0
    osd fsid                  94ff742c-7bfd-4fb5-8dc4-843d10ac6731
    osd id                    3
    osdspec affinity          None
    type                      block
    vdo                       0
    devices                   /dev/vdg
```

2. **获取卷组信息：**

示例

```
[root@host01 ~]# vgs

VG                                #PV #LV #SN Attr   VSize    VFree
db-test                             1   1   0 wz--n- <200.00g <160.00g
test                                1   1   0 wz--n- <200.00g <170.00g
```

3. **停止 Ceph OSD 服务：**

示例

```
[root@host01 ~]# systemctl stop host01a6112da-ed05-11ee-bacd-525400565cda@osd.3.service
```

4. **重新调整逻辑卷的大小、缩小和扩展：**

示例

```
[root@host01 ~]# lvresize -l 100%FREE /dev/db-test/db1
Size of logical volume db-test/db1 changed from 40.00 GiB (10240 extents) to <160.00 GiB (40959 extents).
Logical volume db-test/db1 successfully resized.
```

5. 启动 **cephadm shell：**

语法

```
cephadm shell -m /var/lib/ceph/CLUSTER_FSID/osd.OSD_ID:/var/lib/ceph/osd/ceph-OSD_ID:z
```

示例

```
[root@host01 ~]# cephadm shell -m /var/lib/ceph/1a6112da-ed05-11ee-bacd-
525400565cda/osd.3:/var/lib/ceph/osd/ceph-3:z
```

ceph-bluestore-tool 需要从 cephadm shell 容器内访问 BlueStore 数据，因此必须绑定挂载。使用 -m 选项使 BlueStore 数据可用。

6. **在扩展前检查 Rocks DB 的大小：**

语法

```
ceph-bluestore-tool show-label --path OSD_DIRECTORY_PATH
```

示例

```
[ceph: root@host01 /]# ceph-bluestore-tool show-label --path /var/lib/ceph/osd/ceph-3/
inferring bluefs devices from bluestore path
{
    "/var/lib/ceph/osd/ceph-3/block": {
        "osd_uuid": "94ff742c-7bfd-4fb5-8dc4-843d10ac6731",
        "size": 32212254720,
        "btime": "2024-04-03T08:34:12.742848+0000",
        "description": "main",
        "bfm_blocks": "7864320",
        "bfm_blocks_per_key": "128",
        "bfm_bytes_per_block": "4096",
        "bfm_size": "32212254720",
        "bluefs": "1",
        "ceph_fsid": "1a6112da-ed05-11ee-bacd-525400565cda",
        "ceph_version_when_created": "ceph version 19.0.0-2493-gd82c9aa1
(d82c9aa17f09785fe698d262f9601d87bb79f962) squid (dev)",
        "created_at": "2024-04-03T08:34:15.637253Z",
        "elastic_shared_blobs": "1",
        "kv_backend": "rocksdb",
        "magic": "ceph osd volume v026",
        "mkfs_done": "yes",
        "osd_key": "AQCEFA1m9xuwABAAwKEHkASVbgB1GVt5jYC2Sg==",
        "osdspec_affinity": "None",
        "ready": "ready",
```

```
        "require_osd_release": "19",
        "whoami": "3"
    },
    "/var/lib/ceph/osd/ceph-3/block.db": {
        "osd_uuid": "94ff742c-7bfd-4fb5-8dc4-843d10ac6731",
        "size": 40794497536,
        "btime": "2024-04-03T08:34:12.748816+0000",
        "description": "bluefs db"
    }
}
```

7. **扩展 BlueStore 设备：**

语法

```
ceph-bluestore-tool bluefs-bdev-expand --path OSD_DIRECTORY_PATH
```

示例

```
[ceph: root@host01 /]# ceph-bluestore-tool bluefs-bdev-expand --path
/var/lib/ceph/osd/ceph-3/
inferring bluefs devices from bluestore path
1 : device size 0x27ffbfe000 : using 0x2300000(35 MiB)
2 : device size 0x780000000 : using 0x52000(328 KiB)
Expanding DB/WAL...
1 : expanding  to 0x171794497536
1 : size label updated to 171794497536
```

8. **验证 block.db 是否已扩展：**

语法

```
ceph-bluestore-tool show-label --path OSD_DIRECTORY_PATH
```

示例

```
[ceph: root@host01 /]# ceph-bluestore-tool show-label --path /var/lib/ceph/osd/ceph-3/
inferring bluefs devices from bluestore path
{
    "/var/lib/ceph/osd/ceph-3/block": {
        "osd_uuid": "94ff742c-7bfd-4fb5-8dc4-843d10ac6731",
        "size": 32212254720,
        "btime": "2024-04-03T08:34:12.742848+0000",
        "description": "main",
        "bfm_blocks": "7864320",
        "bfm_blocks_per_key": "128",
        "bfm_bytes_per_block": "4096",
        "bfm_size": "32212254720",
        "bluefs": "1",
        "ceph_fsid": "1a6112da-ed05-11ee-bacd-525400565cda",
        "ceph_version_when_created": "ceph version 19.0.0-2493-gd82c9aa1
(d82c9aa17f09785fe698d262f9601d87bb79f962) squid (dev)",
        "created_at": "2024-04-03T08:34:15.637253Z",
        "elastic_shared_blobs": "1",
        "kv_backend": "rocksdb",
        "magic": "ceph osd volume v026",
        "mkfs_done": "yes",
        "osd_key": "AQCEFA1m9xuwABAAwKEHkASVbgB1GVt5jYC2Sg==",
        "osdspec_affinity": "None",
        "ready": "ready",
        "require_osd_release": "19",
        "whoami": "3"
    },
    "/var/lib/ceph/osd/ceph-3/block.db": {
        "osd_uuid": "94ff742c-7bfd-4fb5-8dc4-843d10ac6731",
        "size": 171794497536,
        "btime": "2024-04-03T08:34:12.748816+0000",
        "description": "bluefs db"
    }
}
```

9. **退出 shell 并重启 OSD：**

示例

```
[root@host01 ~]# systemctl start host01a6112da-ed05-11ee-bacd-
525400565cda@osd.3.service
osd.3          host01          running (15s)    0s ago  13m    46.9M    4096M  19.0.0-2493-
gd82c9aa1  3714003597ec  02150b3b6877
```

7.10. 将批处理模式与 CEPH-VOLUME搭配使用

当提供单一设备时，batch 子命令可自动创建多个 OSD。

ceph-volume 命令根据驱动器类型决定使用创建 OSD 的最佳方法。Ceph OSD 优化取决于可用的设备：

- **如果所有设备都是传统的硬盘驱动器，batch 会为每个设备创建一个 OSD。**
- **如果所有设备都是固态硬盘，则 batch 会为每个设备创建两个 OSD。**
- **如果混合使用传统硬盘驱动器和固态驱动器，batch 使用传统的硬盘驱动器用于数据，并在固态硬盘上创建最大可能的日志(block.db)。**

注意

batch 子命令不支持为 write-ahead-log (block.wal) 设备创建单独的逻辑卷。

先决条件

- **一个正在运行的 Red Hat Ceph Storage 集群。**
- **对 Ceph OSD 节点的 root 级别访问权限。**

流程

1. 在几个驱动器中创建 OSD：

语法

```
ceph-volume lvm batch --bluestore PATH_TO_DEVICE [PATH_TO_DEVICE]
```

示例

```
[ceph: root@host01 /]# ceph-volume lvm batch --bluestore /dev/sda /dev/sdb /dev/nvme0n1
```

其它资源

- 如需更多详细信息，请参阅 *Red Hat Ceph Storage 管理指南*中的*使用 'ceph-volume' 创建 Ceph OSD*。

7.11. 使用 CEPH-VOLUME进行数据

zap 子命令从逻辑卷或分区中删除所有数据和文件系统。

您可以使用 zap 子命令来 zap 逻辑卷、分区或 Ceph OSD 使用的原始设备来重复使用。删除给定逻辑卷或分区上的任何文件系统都会被删除，且所有数据都会被清除。

另外，您还可以使用 --destroy 标志完成逻辑卷、分区或者物理设备删除。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。

- 对 **Ceph OSD** 节点的 **root** 级别访问权限。

流程

- **zap** 逻辑卷：

  语法

  ```
  ceph-volume lvm zap VOLUME_GROUP_NAME/LOGICAL_VOLUME_NAME [--destroy]
  ```

  示例

  ```
  [ceph: root@host01 /]# ceph-volume lvm zap osd-vg/data-lv
  ```

- **zap** 分区：

  语法

  ```
  ceph-volume lvm zap DEVICE_PATH_PARTITION [--destroy]
  ```

  示例

  ```
  [ceph: root@host01 /]# ceph-volume lvm zap /dev/sdc1
  ```

- **zap 原始设备：**

语法

```
ceph-volume lvm zap DEVICE_PATH --destroy
```

示例

```
[ceph: root@host01 /]# ceph-volume lvm zap /dev/sdc --destroy
```

- **清除具有 OSD ID 的多个设备：**

语法

```
ceph-volume lvm zap --destroy --osd-id OSD_ID
```

示例

```
[ceph: root@host01 /]# ceph-volume lvm zap --destroy --osd-id 16
```

注意

所有相对设备都是 zapped。

- **使用 FSID 清除 OSD：**

语法

```
ceph-volume lvm zap --destroy --osd-fsid OSD_FSID
```

示例

```
[ceph: root@host01 /]# ceph-volume lvm zap --destroy --osd-fsid 65d7b6b1-e41a-4a3c-b363-83ade63cb32b
```

注意

所有相对设备都是 zapped。

第 8 章 CEPH 性能基准

作为存储管理员，您可以对 Red Hat Ceph Storage 集群的基准测试性能进行基准测试。本节的目的是让 Ceph 管理员能够了解 Ceph 的原生基准工具。这些工具将深入探讨 Ceph 存储集群的运行情况。这不是 Ceph 性能基准的确定指南，也不是一个有关如何相应地调整 Ceph 的指南。

8.1. 性能基准

OSD（包括日志、磁盘和网络吞吐量）应具有一个用于比较的性能基准。您可以通过将基准性能数据与 Ceph 原生工具中的数据进行比较来识别潜在的调优机会。Red Hat Enterprise Linux 具有许多内置工具，以及开源社区工具的 plethora，可用于帮助完成这些任务。

其它资源

- 有关一些可用工具的详情，请查看知识库文章。

8.2. CEPH 性能基准

Ceph 包含 rados bench 命令，用于在 RADOS 存储群集上执行性能基准测试。命令将执行写入测试，以及两种类型的读测试。在测试读取和写入性能时，--no-cleanup 选项非常重要。默认情况下，rados bench 命令会删除它写入存储池的对象。保留这些对象后，可以使用两个读取测试来测量顺序读取和随机读取的性能。

注意

在运行这些性能测试前，运行以下命令丢弃所有文件系统缓存：

示例

```
[ceph: root@host01 /]# echo 3 | sudo tee /proc/sys/vm/drop_caches && sudo sync
```

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。

- **节点的根级别访问权限。**

流程

1. **创建新存储池：**

示例

```
[ceph: root@host01 /]# ceph osd pool create testbench 100 100
```

2. **对新创建的存储池执行 10 秒的写入测试：**

示例

```
[ceph: root@host01 /]# rados bench -p testbench 10 write --no-cleanup

Maintaining 16 concurrent writes of 4194304 bytes for up to 10 seconds or 0 objects
 Object prefix: benchmark_data_cephn1.home.network_10510
   sec Cur ops   started  finished  avg MB/s  cur MB/s  last lat   avg lat
     0       0         0         0         0         0         -         0
     1      16        16         0         0         0         -         0
     2      16        16         0         0         0         -         0
     3      16        16         0         0         0         -         0
     4      16        17         1  0.998879         1   3.19824   3.19824
     5      16        18         2   1.59849         4   4.56163   3.87993
     6      16        18         2   1.33222         0         -   3.87993
     7      16        19         3   1.71239         2   6.90712     4.889
     8      16        25         9   4.49551        24   7.75362   6.71216
     9      16        25         9   3.99636         0         -   6.71216
    10      16        27        11   4.39632         4   9.65085   7.18999
    11      16        27        11   3.99685         0         -   7.18999
    12      16        27        11   3.66397         0         -   7.18999
    13      16        28        12   3.68975   1.33333   12.8124   7.65853
    14      16        28        12   3.42617         0         -   7.65853
    15      16        28        12   3.19785         0         -   7.65853
    16      11        28        17   4.24726   6.66667   12.5302   9.27548
    17      11        28        17   3.99751         0         -   9.27548
    18      11        28        17   3.77546         0         -   9.27548
    19      11        28        17   3.57683         0         -   9.27548
 Total time run:         19.505620
```

```
Total writes made:      28
Write size:             4194304
Bandwidth (MB/sec):     5.742

Stddev Bandwidth:       5.4617
Max bandwidth (MB/sec): 24
Min bandwidth (MB/sec): 0
Average Latency:        10.4064
Stddev Latency:         3.80038
Max latency:            19.503
Min latency:            3.19824
```

3. **为存储池执行一次10 秒的连续读测试 ：**

示例

```
[ceph: root@host01 /]# rados bench -p testbench 10 seq

sec Cur ops   started  finished  avg MB/s  cur MB/s  last lat   avg lat
  0       0         0         0         0         0         -         0
Total time run:        0.804869
Total reads made:      28
Read size:             4194304
Bandwidth (MB/sec):    139.153

Average Latency:       0.420841
Max latency:           0.706133
Min latency:           0.0816332
```

4. **为存储池执行一次 10 秒的随机读取测试 ：**

示例

```
[ceph: root@host01 /]# rados bench -p testbench 10 rand

sec Cur ops   started  finished  avg MB/s  cur MB/s  last lat   avg lat
  0       0         0         0         0         0         -         0
  1      16        46        30   119.801       120  0.440184  0.388125
```

```
  2      16       81       65   129.408       140  0.577359  0.417461
  3      16      120      104   138.175       156  0.597435  0.409318
  4      15      157      142   141.485       152  0.683111  0.419964
  5      16      206      190   151.553       192  0.310578  0.408343
  6      16      253      237   157.608       188 0.0745175  0.387207
  7      16      287      271   154.412       136  0.792774   0.39043
  8      16      325      309   154.044       152  0.314254   0.39876
  9      16      362      346   153.245       148  0.355576  0.406032
 10      16      405      389   155.092       172   0.64734  0.398372
Total time run:       10.302229
Total reads made:     405
Read size:            4194304
Bandwidth (MB/sec):    157.248

Average Latency:      0.405976
Max latency:          1.00869
Min latency:          0.0378431
```

5.

要增加并发读取和写入的数量，请使用 -t 选项，默认为 16 个线程。另外，-b 参数可以调整所写入对象的大小。默认对象大小为 4 MB。安全最大对象大小为 16 MB。红帽建议将这些基准测试中的多个副本运行到不同的池。这样做显示与多个客户端的性能更改。

添加 --run-name *LABEL* 选项以控制在基准测试过程中写入的对象名称。通过更改每个运行命令实例的 --run-name 标签，可以同时运行多个 rados bench 命令。这可防止当多个客户端试图访问同一对象时可能会出现潜在的 I/O 错误，并允许不同的客户端访问不同的对象。在尝试模拟真实工作负载时，--run-name 选项也很有用。

示例

```
[ceph: root@host01 /]# rados bench -p testbench 10 write -t 4 --run-name client1

Maintaining 4 concurrent writes of 4194304 bytes for up to 10 seconds or 0 objects
 Object prefix: benchmark_data_node1_12631
   sec Cur ops   started  finished  avg MB/s  cur MB/s  last lat   avg lat
     0       0         0         0         0         0         -         0
     1       4         4         0         0         0         -         0
     2       4         6         2   3.99099         4   1.94755   1.93361
     3       4         8         4   5.32498         8     2.978   2.44034
     4       4         8         4   3.99504         0         -   2.44034
     5       4        10         6   4.79504         4   2.92419    2.4629
     6       3        10         7   4.64471         4   3.02498    2.5432
     7       4        12         8   4.55287         4   3.12204   2.61555
     8       4        14        10    4.9821         8   2.55901   2.68396
     9       4        16        12   5.31621         8   2.68769   2.68081
    10       4        17        13   5.18488         4   2.11937   2.63763
    11       4        17        13   4.71431         0         -   2.63763
```

```
   12      4       18       14   4.65486        2   2.4836   2.62662
   13      4       18       14   4.29757        0        -   2.62662
Total time run:         13.123548
Total writes made:      18
Write size:             4194304
Bandwidth (MB/sec):     5.486

Stddev Bandwidth:       3.0991
Max bandwidth (MB/sec): 8
Min bandwidth (MB/sec): 0
Average Latency:        2.91578
Stddev Latency:         0.956993
Max latency:            5.72685
Min latency:            1.91967
```

6. **删除 rados bench 命令创建的数据：**

示例

```
[ceph: root@host01 /]# rados -p testbench cleanup
```

8.3. 基准测试 CEPH 块性能

Ceph 包含 rbd bench-write 命令，以测试对块设备测量吞吐量和延迟情况的连续写入。默认字节大小为 4096，默认 I/O 线程数为 16，默认的写入字节数为 1 GB。这些默认值可通过 --io-size, --io-threads 和 --io-total 选项分别进行修改。

先决条件

- **一个正在运行的 Red Hat Ceph Storage 集群。**
- **节点的根级别访问权限。**

流程

- 对块设备执行写入性能测试

示例

```
[root@host01 ~]# rbd bench --io-type write image01 --pool=testbench

bench-write  io_size 4096 io_threads 16 bytes 1073741824 pattern seq
  SEC       OPS   OPS/SEC   BYTES/SEC
    2     11127   5479.59  22444382.79
    3     11692   3901.91  15982220.33
    4     12372   2953.34  12096895.42
    5     12580   2300.05  9421008.60
    6     13141   2101.80  8608975.15
    7     13195    356.07  1458459.94
    8     13820    390.35  1598876.60
    9     14124    325.46  1333066.62
    ..
```

其它资源

- 有关 rbd 命令的更多信息，请参阅 ***Red Hat Ceph Storage Block Device Guide*** 中的 *Ceph 块设备* 一章。

8.4. CEPHFS 性能基准测试

您可以使用 FIO 工具对 Ceph 文件系统(CephFS)性能进行基准测试。此工具也可用于对 Ceph 块设备进行基准测试。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。
- 节点的根级别访问权限。
- 在节点上安装了 FIO 工具。如需了解更多详细信息，请参阅 KCS *如何安装 Flexible I/O Tester (fio)性能基准工具*。

- 挂载到节点上的块设备或 Ceph 文件系统。

流程

1. 导航到挂载了 Block Device 或 CephFS 的节点或应用程序：

   示例

   ```
   [root@host01 ~]# cd /mnt/ceph-block-device
   [root@host01 ~]# cd /mnt/ceph-file-system
   ```

2. 运行 FIO 命令。从 4k 开始 bs 值，并在 2 个增量(4k、8k、16k、32k … 128k… 512k, 1m, 2m, 4m)以及不同的 iodepth 设置重复。您还应该在预期的工作负载操作大小上运行测试。

   具有不同 iodepth 值的 4K 测试示例

   ```
   fio --name=randwrite --rw=randwrite --direct=1 --ioengine=libaio --bs=4k --iodepth=32 --
   size=5G --runtime=60 --group_reporting=1
   ```

   具有不同 iodepth 值的 8K 测试示例

   ```
   fio --name=randwrite --rw=randwrite --direct=1 --ioengine=libaio --bs=8k --iodepth=32 --
   size=5G --runtime=60 --group_reporting=1
   ```

注意

有关使用 fio 命令的详情，请参考 fio 手册页。

8.5. CEPH 对象网关性能基准测试

您可以使用 s3cmd 工具对 Ceph 对象网关性能进行基准测试。

使用 get 和 put 请求来确定性能。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。
- 节点的根级别访问权限。
- 在节点上安装的 s3cmd。

流程

1. 上传文件并测量速度。time 命令测量上传持续时间。

   语法

   ```
   time s3cmd put PATH_OF_SOURCE_FILE PATH_OF_DESTINATION_FILE
   ```

   示例

   ```
   time s3cmd put /path-to-local-file s3://bucket-name/remote/file
   ```

使用您要上传的文件替换 /path-to-local-file，将 s3://bucket-name/remote/file 替换为 S3 存储桶中的目的地。

2. 下载文件并测量速度。time 命令测量下载持续时间。

语法

```
time s3cmd get PATH_OF_DESTINATION_FILE DESTINATION_PATH
```

示例

```
time s3cmd get s3://bucket-name/remote/file /path-to-local-destination
```

使用您要下载的 S3 对象替换 s3://bucket-name/remote/file，使用您要保存文件的本地目录替换 /path-to-local-destination。

3. 列出指定存储桶中的所有对象，并测量响应时间。

语法

```
time s3cmd ls s3://BUCKET_NAME
```

示例

```
time s3cmd ls s3://bucket-name
```

4. **分析输出，以计算上传/下载速度，并根据 time 命令报告的持续时间来测量响应时间。**

第 9 章 CEPH 性能计数器

作为存储管理员，您可以收集 Red Hat Ceph Storage 集群的性能指标。Ceph 性能计数器是内部基础架构指标的集合。此指标数据的集合、聚合和图表可以通过一组工具进行，并可用于性能分析。

9.1. 访问 CEPH 性能计数器

性能计数器可通过 Ceph 监控和 OSD 的套接字接口提供。每个对应守护进程的套接字文件默认位于 /var/run/ceph 下。性能计数器分组到集合名称中。这些集合名称代表子系统或子系统实例。

以下是 monitor 和 OSD 集合名称类别的完整列表，其中包含每个的简短描述信息：

Monitor 集合名称目录

- Cluster Metrics - 显示存储集群的信息：监控、OSD、池和 PG
- Level Database Metrics - 显示后端 KeyValueStore 数据库的信息
- Monitor Metrics - 显示常规监控信息
- Paxos Metrics - 显示集群仲裁管理的信息
- Throttle Metrics - 显示监控器节流的统计信息

OSD 集合名称目录

- Write Back Throttle Metrics - 显示写入后节流的统计是如何跟踪未清空的 IO
- Level Database Metrics - 显示后端 KeyValueStore 数据库的信息
- Objecter Metrics - 显示各种基于对象的操作的信息

- 读和写操作指标 - 显示各种读写操作的信息
- 恢复状态指标 - 显示 - 显示各种恢复状态的延迟
- OSD Throttle Metrics - 显示 OSD 节流的统计

RADOS 网关集合名称 Categories

- Object Gateway Client Metrics - 显示 GET 和 PUT 请求的统计信息
- Objecter Metrics - 显示各种基于对象的操作的信息
- Object Gateway Throttle Metrics - 显示 OSD 节流方式的统计信息

9.2. 显示 CEPH 性能计数器

ceph daemon *DAEMON_NAME* perf schema 命令输出可用的指标。每个指标都有一个关联的位字段值类型。

先决条件

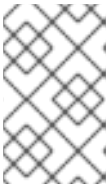
- 一个正在运行的 Red Hat Ceph Storage 集群。
- 节点的根级别访问权限。

流程

1. 查看指标的 schema：

Synatx

```
ceph daemon DAEMON_NAME perf schema
```



注意

您必须从节点运行 ceph daemon 命令来运行守护进程。

2. **从监控节点执行 ceph daemon *DAEMON_NAME* perf schema 命令：**

示例

```
[ceph: root@host01 /]# ceph daemon mon.host01 perf schema
```

3. **从 OSD 节点执行 ceph daemon *DAEMON_NAME* perf schema 命令：**

示例

```
[ceph: root@host01 /]# ceph daemon osd.11 perf schema
```

表 9.1. 位字段值定义

位	含义
1	浮点值
2	未签名的 64 位整数值
4	平均(Sum + Count)
8	计数

每个值都将有位 1 或 2 设置来表示类型，可以是浮点或整数值。当位 4 被设置时，将有两个值可供读取，分别是 sum 和 count。当设置了第 8 位时，以前间隔的平均间隔值为总 delta 值（自上一次读取以来）除以 delta 的数量。另外，从右边的值分离会提供生命周期平均值值。通常，这用于衡量延迟、请求数和请求延迟总和。一些位的值会被合并，如 5、6 和 10。位值 5 是位 1 和位 4 的组合。这意味着平均值将是一个浮点值。位值 6 是位 2 和位 4 的组合。这意味着平均值为一个整数。位值 10 是位 2 和位 8 的组合。这意味着计数器值将是整数值。

其它资源

- 如需更多详细信息，请参阅 *Red Hat Ceph Storage 管理指南*中的*平均数和总和*部分。

9.3. 转储 CEPH 性能计数器

ceph daemon .. perf dump 命令会输出当前值，并将指标分组到各个子系统的集合名称下。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。
- 节点的根级别访问权限。

流程

1. 查看当前的指标数据：

语法

```
ceph daemon DAEMON_NAME perf dump
```

注意

您必须从节点运行 ceph daemon 命令来运行守护进程。

2. 从 monitor 节点执行 ceph daemon .. perf dump 命令：

```
[ceph: root@host01 /]# ceph daemon mon.host01 perf dump
```

3. 从 OSD 节点执行 ceph daemon .. perf dump 命令：

```
[ceph: root@host01 /]# ceph daemon osd.11 perf dump
```

其它资源

- 要查看每个可用的 monitor 指标的简短描述，请参阅 *Ceph 监控指标表*。

9.4. 平均计数和总和

所有延迟数量都有一个 bit 字段值 5。此字段包含平均计数和 sum 的浮点值。avgcount 是这个范围内的操作数，sum 是总延迟（以秒为单位）。将 sum 除以 avgcount 的值为您提供了关于每个操作的延迟理念。

其它资源

- 要查看每个 OSD 指标的简短描述，请参见 *Ceph OSD 表*。

9.5. CEPH 监控指标

- *集群指标表*
- *级别数据库指标表*
- *常规监控指标表*
- *Paxos 指标表*
- *Throttle 指标表*

表 9.2. 集群指标表

集合名称	指标名称	位字段值	简短描述
cluster	**num_mon**	2	监控器数
	num_mon_quorum	2	仲裁中的 monitor 数量
	num_osd	2	OSD 的总数
	num_osd_up	2	已启动的 OSD 数量
	num_osd_in	2	集群中的 OSD 数量
	osd_epoch	2	OSD map 的当前 epoch
	osd_bytes	2	集群总容量（以字节为单位）
	osd_bytes_used	2	集群中的已用字节数
	osd_bytes_avail	2	集群中的可用字节数
	num_pool	2	池数
	num_pg	2	放置组总数
	num_pg_active_clean	2	active+clean 状态的放置组数量
	num_pg_active	2	处于活跃状态的放置组数量
	num_pg_peering	2	处于 peering 状态的放置组数量
	num_object	2	集群中的对象总数
	num_object_degraded	2	降级数量（减少副本）对象
	num_object_misplaced	2	对象中错误的原位（集群位置）数
	num_object_unfound	2	未找到的对象数量
	num_bytes	2	所有对象的字节数
	num_mds_up	2	启动的 MDS 数量

集合名称	指标名称	位字段值	简短描述
	num_mds_in	2	集群中的 MDS 数量
	num_mds_failed	2	失败的 MDS 数量
	mds_epoch	2	MDS 映射的当前 epoch

表 9.3. 级别数据库指标表

集合名称	指标名称	位字段值	简短描述
leveldb	**leveldb_get**	10	Gets
	leveldb_transaction	10	Transactions
	leveldb_compact	10	Compactions
	leveldb_compact_range	10	按范围完成
	leveldb_compact_queue_merge	10	在压缩队列中合并范围
	leveldb_compact_queue_len	2	压缩队列长度

表 9.4. 常规监控指标表

集合名称	指标名称	位字段值	简短描述
mon	**num_sessions**	2	当前打开的 monitor 会话数量
	session_add	10	创建的 monitor 会话数量
	session_rm	10	monitor 中的 remove_session 调用数量
	session_trim	10	修剪监控会话的数量
	num_elections	10	参与的 elections monitor 数量
	election_call	10	有 monitor 启动的选举数
	election_win	10	监控可享受的选举数量
	election_lose	10	监控丢失的选举数量

集合名称	指标名称	位字段值	简短描述

表 9.5. Paxos Metrics Table

集合名称	指标名称	位字段值	简短描述
paxos	**start_leader**	10	以领导角色开始
	start_peon	10	以 peon 角色启动
	restart	10	重启
	refresh	10	刷新
	refresh_latency	5	刷新延迟
	begin	10	启动和处理开始
	begin_keys	6	开始时事务中的键
	begin_bytes	6	开始时事务中的数据
	begin_latency	5	开始操作的延迟
	commit	10	提交
	commit_keys	6	提交时事务中的键
	commit_bytes	6	提交时事务中的数据
	commit_latency	5	提交延迟
	collect	10	peon 收集
	collect_keys	6	peon collect 时事务中的键
	collect_bytes	6	peon collect 时事务中的数据
	collect_latency	5	peon 收集延迟
	collect_uncommitted	10	在开始的和处理的 collect 中未提交的值
	collect_timeout	10	收集超时

集合名称	指标名称	位字段值	简短描述
	accept_timeout	10	接受超时
	lease_ack_timeout	10	租期确认超时
	lease_timeout	10	租期超时
	store_state	10	在磁盘上存储共享状态
	store_state_keys	6	存储在事务中的密钥
	store_state_bytes	6	存储在存储状态下的事务中的数据
	store_state_latency	5	存储状态延迟
	share_state	10	状态共享
	share_state_keys	6	共享状态的密钥
	share_state_bytes	6	处于共享状态的数据
	new_pn	10	新提议号查询
	new_pn_latency	5	新的提议号获得延迟

表 9.6. throttle Metrics Table

集合名称	指标名称	位字段值	简短描述
throttle-*	**val**	10	目前可用节流
	max	10	最大值 throttle
	get	10	Gets
	get_sum	10	获取数据
	get_or_fail_fail	10	在 get_or_fail 时被阻断
	get_or_fail_success	10	在 get_or_fail 期间获得成功
	take	10	Takes
	take_sum	10	获取的数据

集合名称	指标名称	位字段值	简短描述
	put	10	Puts
	put_sum	10	放置数据
	wait	5	等待延迟

9.6. CEPH OSD 指标

- *写回 Throttle Metrics Table*
- *级别数据库指标表*
- *Objecter Metrics Table*
- *读和写操作指标表*
- *恢复状态指标表*
- *OSD Throttle Metrics Table*

表 9.7. 写回 Throttle Metrics Table

集合名称	指标名称	位字段值	简短描述
WBThrottle	**bytes_dirtied**	2	脏数据
	bytes_wb	2	写入数据
	ios_dirtied	2	脏操作
	ios_wb	2	写入操作
	inodes_dirtied	2	等待写入的条目
	inodes_wb	2	写入条目

表 9.8. 级别数据库指标表

集合名称	指标名称	位字段值	简短描述
leveldb	**leveldb_get**	10	Gets
	leveldb_transaction	10	Transactions
	leveldb_compact	10	Compactions
	leveldb_compact_range	10	按范围完成
	leveldb_compact_queue_merge	10	在压缩队列中合并范围
	leveldb_compact_queue_len	2	压缩队列长度

表 9.9. Objecter Metrics Table

集合名称	指标名称	位字段值	简短描述
objecter	**op_active**	2	活跃操作
	op_laggy	2	Laggy 操作
	op_send	10	发送的操作
	op_send_bytes	10	发送的数据
	op_resend	10	重新发送操作
	op_ack	10	提交回调
	op_commit	10	操作提交
	op	10	操作
	op_r	10	读取操作
	op_w	10	写操作
	op_rmw	10	Read-modify-write 操作
	op_pg	10	PG 操作

集合名称	指标名称	位字段值	简短描述
	osdop_stat	10	Stat 操作
	osdop_create	10	创建对象操作
	osdop_read	10	读取操作
	osdop_write	10	写操作
	osdop_writefull	10	编写完整对象操作
	osdop_append	10	附加操作
	osdop_zero	10	将对象设置为零操作
	osdop_truncate	10	截断对象操作
	osdop_delete	10	删除对象操作
	osdop_mapext	10	映射扩展操作
	osdop_sparse_read	10	稀疏读取操作
	osdop_clonerange	10	克隆范围操作
	osdop_getxattr	10	Get xattr 操作
	osdop_setxattr	10	设置 xattr 操作
	osdop_cmpxattr	10	xattr 比较操作
	osdop_rmxattr	10	删除 xattr 操作
	osdop_resetxattrs	10	重置 xattr 操作
	osdop_tmap_up	10	TMAP 更新操作
	osdop_tmap_put	10	TMAP put 操作
	osdop_tmap_get	10	TMAP get 操作
	osdop_call	10	调用（执行）操作
	osdop_watch	10	按对象操作监视
	osdop_notify	10	通知对象操作

集合名称	指标名称	位字段值	简短描述
	osdop_src_cmpxattr	10	多操作中的扩展属性比较
	osdop_other	10	其他操作
	linger_active	2	活跃的闲置操作
	linger_send	10	发送的闲置操作
	linger_resend	10	重新闲置操作
	linger_ping	10	将 ping 发送到闲置操作
	poolop_active	2	活跃池操作
	poolop_send	10	发送池操作
	poolop_resend	10	重组池操作
	poolstat_active	2	Active get pool stat 操作
	poolstat_send	10	池 stat 操作发送
	poolstat_resend	10	重新设置池统计
	statfs_active	2	statfs 操作
	statfs_send	10	发送的 FS stats
	statfs_resend	10	重新发送的 FS stats
	command_active	2	活跃命令
	command_send	10	发送命令
	command_resend	10	重新发送命令
	map_epoch	2	OSD map epoch
	map_full	10	收到的完整 OSD 映射
	map_inc	10	接收的增量 OSD map
	osd_sessions	2	开放会话
	osd_session_open	10	会话已打开

集合名称	指标名称	位字段值	简短描述
	osd_session_close	10	会话关闭
	osd_laggy	2	Laggy OSD 会话

表 9.10. 读和写操作指标表

集合名称	指标名称	位字段值	简短描述
osd	**op_wip**	2	复制当前正在被处理的操作（主）
	op_in_bytes	10	客户端操作总写入大小
	op_out_bytes	10	客户端操作总读取大小
	op_latency	5	客户端操作的延迟（包括队列时间）
	op_process_latency	5	客户端操作的延迟（不包括队列时间）
	op_r	10	客户端读取操作
	op_r_out_bytes	10	读取客户端数据
	op_r_latency	5	读取操作的延迟（包括队列时间）
	op_r_process_latency	5	读取操作的延迟（不包括队列时间）
	op_w	10	客户端写入操作
	op_w_in_bytes	10	写入的客户端数据
	op_w_rlat	5	客户端写入操作可读/应用延迟
	op_w_latency	5	写入操作的延迟（包括队列时间）
	op_w_process_latency	5	写入操作的延迟（不包括队列时间）
	op_rw	10	客户端 read-modify-write 操作

集合名称	指标名称	位字段值	简短描述
	op_rw_in_bytes	10	客户端 read-modify-write 操作写入
	op_rw_out_bytes	10	客户端 read-modify-write 操作读出
	op_rw_rlat	5	客户端 read-modify-write 操作可读/应用延迟
	op_rw_latency	5	读写操作的延迟（包括队列时间）
	op_rw_process_latency	5	读写操作的延迟（不包括队列时间）
	subop	10	Suboperations
	subop_in_bytes	10	Suboperations 总数
	subop_latency	5	Suboperations 延迟
	subop_w	10	复制写入
	subop_w_in_bytes	10	复制的写入数据大小
	subop_w_latency	5	复制的写入延迟
	subop_pull	10	Suboperations pull 请求
	subop_pull_latency	5	Suboperations pull 延迟
	subop_push	10	Suboperations push 消息
	subop_push_in_bytes	10	Suboperations 推送的大小
	subop_push_latency	5	Suboperations push 延迟
	pull	10	发送的拉取请求
	push	10	推送发送的消息
	push_out_bytes	10	推送的大小

集合名称	指标名称	位字段值	简短描述
	push_in	10	入站推送消息
	push_in_bytes	10	入站推送的大小
	recovery_ops	10	开始恢复操作
	loadavg	2	CPU 负载
	buffer_bytes	2	分配的缓冲大小总量
	numpg	2	放置组
	numpg_primary	2	此 osd 是主的放置组
	numpg_replica	2	此 osd 是副本的放置组
	numpg_stray	2	准备好从此 osd 删除 PG
	heartbeat_to_peers	2	发送给对等点的心跳(ping)
	heartbeat_from_peers	2	接收来自其中的心跳(ping)对等点
	map_messages	10	OSD map 消息
	map_message_epochs	10	OSD map epochs
	map_message_epoch_dups	10	OSD map 重复
	stat_bytes	2	OSD 大小
	stat_bytes_used	2	使用的空间
	stat_bytes_avail	2	可用空间
	copyfrom	10	RADOS 'copy-from' 操作
	tier_promote	10	等级提升
	tier_flush	10	Tier flushes
	tier_flush_fail	10	失败的分层清除

集合名称	指标名称	位字段值	简短描述
	tier_try_flush	10	tier flush 尝试
	tier_try_flush_fail	10	失败的分层清除尝试
	tier_evict	10	等级驱除
	tier_whiteout	10	Tier whiteouts
	tier_dirty	10	设定脏层标志
	tier_clean	10	清理脏层标志
	tier_delay	10	Tier delays (agent waiting)
	tier_proxy_read	10	层代理读取
	agent_wake	10	分层代理唤醒
	agent_skip	10	代理跳过的对象
	agent_flush	10	分层代理清除
	agent_evict	10	分层代理驱除
	object_ctx_cache_hit	10	对象上下文缓存命
	object_ctx_cache_total	10	对象上下文缓存查找
	ceph_cluster_osd_blocklist_count	2	阻塞的客户端数量

表 9.11. 恢复状态指标表

集合名称	指标名称	位字段值	简短描述
recoverystate_perf	**initial_latency**	5	初始恢复状态延迟
	started_latency	5	开始恢复状态延迟
	reset_latency	5	重置恢复状态延迟
	start_latency	5	启动恢复状态延迟

集合名称	指标名称	位字段值	简短描述
	primary_latency	5	主要恢复状态延迟
	peering_latency	5	对等恢复状态延迟
	backfilling_latency	5	回填恢复状态延迟
	waitremotebackfillreserved_latency	5	等待远程回填保留恢复状态延迟
	waitlocalbackfillreserved_latency	5	等待本地回填保留恢复状态延迟
	notbackfilling_latency	5	Notbackfilling 恢复状态延迟
	repnotrecovering_latency	5	重新恢复恢复状态延迟
	repwaitrecoveryreserved_latency	5	rep 等待恢复保留恢复状态延迟
	repwaitbackfillreserved_latency	5	Rep 等待回填保留的恢复状态
	RepRecovering_latency	5	重新恢复恢复状态延迟
	activating_latency	5	激活恢复状态延迟
	waitlocalrecoveryreserved_latency	5	等待本地恢复状态延迟
	waitremoterecoveryreserved_latency	5	等待远程恢复保留状态延迟
	recovering_latency	5	恢复状态延迟
	recovered_latency	5	恢复状态延迟
	clean_latency	5	清理恢复状态延迟
	active_latency	5	主动恢复状态延迟
	replicaactive_latency	5	Replicaactive 恢复状态延迟
	stray_latency	5	stray 恢复状态延迟

集合名称	指标名称	位字段值	简短描述
	getinfo_latency	5	Getinfo 恢复状态延迟
	getlog_latency	5	Getlog 恢复状态延迟
	waitactingchange_latency	5	Waitactingchange 恢复状态延迟
	incomplete_latency	5	恢复状态延迟不完整
	getmissing_latency	5	获取恢复状态延迟
	waitupthru_latency	5	Waitupthru 恢复状态延迟

表 9.12. OSD Throttle Metrics Table

集合名称	指标名称	位字段值	简短描述
throttle-*	**val**	10	目前可用节流
	max	10	最大值 throttle
	get	10	Gets
	get_sum	10	获取数据
	get_or_fail_fail	10	在 get_or_fail 时被阻断
	get_or_fail_success	10	在 get_or_fail 期间获得成功
	take	10	Takes
	take_sum	10	获取的数据
	put	10	Puts
	put_sum	10	放置数据
	wait	5	等待延迟

9.7. CEPH 对象网关指标

- *Ceph 对象网关客户端表*

- *Objecter Metrics Table*
- *Ceph 对象网关 Throttle Metrics Table*

表 9.13. Ceph 对象网关客户端指标表

集合名称	指标名称	位字段值	简短描述
client.rgw.<rgw_node_name>	req	10	Requests
	failed_req	10	中止的请求
	copy_obj_ops	10	复制对象
	copy_obj_bytes	10	复制对象的大小
	copy_obj_lat	10	复制对象延迟
	del_obj_ops	10	删除对象
	del_obj_bytes	10	删除对象的大小
	del_obj_lat	10	删除对象延迟
	del_bucket_ops	10	Delete Buckets
	del_bucket_lat	10	删除存储桶延迟
	get	10	Gets
	get_b	10	gets 的大小
	get_initial_lat	5	获取延迟
	list_obj_ops	10	列出对象
	list_obj_lat	10	列出对象延迟
	list_buckets_ops	10	列出存储桶
	list_buckets_lat	10	列出存储桶延迟
	put	10	Puts

集合名称	指标名称	位字段值	简短描述
	put_b	10	puts 的大小
	put_initial_lat	5	Put 延迟
	qlen	2	队列长度
	qactive	2	活跃请求队列
	cache_hit	10	缓存命中
	cache_miss	10	缓存未命中
	keystone_token_cache_hit	10	Keystone 令牌缓存命
	keystone_token_cache_miss	10	Keystone 令牌缓存未命中

表 9.14. Objecter Metrics Table

集合名称	指标名称	位字段值	简短描述
objecter	**op_active**	2	活跃操作
	op_laggy	2	Laggy 操作
	op_send	10	发送的操作
	op_send_bytes	10	发送的数据
	op_resend	10	重新发送操作
	op_ack	10	提交回调
	op_commit	10	操作提交
	op	10	操作
	op_r	10	读取操作
	op_w	10	写操作
	op_rmw	10	Read-modify-write 操作
	op_pg	10	PG 操作

集合名称	指标名称	位字段值	简短描述
	osdop_stat	10	Stat 操作
	osdop_create	10	创建对象操作
	osdop_read	10	读取操作
	osdop_write	10	写操作
	osdop_writefull	10	编写完整对象操作
	osdop_append	10	附加操作
	osdop_zero	10	将对象设置为零操作
	osdop_truncate	10	截断对象操作
	osdop_delete	10	删除对象操作
	osdop_mapext	10	映射扩展操作
	osdop_sparse_read	10	稀疏读取操作
	osdop_clonerange	10	克隆范围操作
	osdop_getxattr	10	Get xattr 操作
	osdop_setxattr	10	设置 xattr 操作
	osdop_cmpxattr	10	xattr 比较操作
	osdop_rmxattr	10	删除 xattr 操作
	osdop_resetxattrs	10	重置 xattr 操作
	osdop_tmap_up	10	TMAP 更新操作
	osdop_tmap_put	10	TMAP put 操作
	osdop_tmap_get	10	TMAP get 操作
	osdop_call	10	调用（执行）操作
	osdop_watch	10	按对象操作监视
	osdop_notify	10	通知对象操作

集合名称	指标名称	位字段值	简短描述
	osdop_src_cmpxattr	10	多操作中的扩展属性比较
	osdop_other	10	其他操作
	linger_active	2	活跃的闲置操作
	linger_send	10	发送的闲置操作
	linger_resend	10	重新闲置操作
	linger_ping	10	将 ping 发送到闲置操作
	poolop_active	2	活跃池操作
	poolop_send	10	发送池操作
	poolop_resend	10	重组池操作
	poolstat_active	2	Active get pool stat 操作
	poolstat_send	10	池 stat 操作发送
	poolstat_resend	10	重新设置池统计
	statfs_active	2	statfs 操作
	statfs_send	10	发送的 FS stats
	statfs_resend	10	重新发送的 FS stats
	command_active	2	活跃命令
	command_send	10	发送命令
	command_resend	10	重新发送命令
	map_epoch	2	OSD map epoch
	map_full	10	收到的完整 OSD 映射
	map_inc	10	接收的增量 OSD map
	osd_sessions	2	开放会话
	osd_session_open	10	会话已打开

集合名称	指标名称	位字段值	简短描述
	osd_session_close	10	会话关闭
	osd_laggy	2	Laggy OSD 会话

表 9.15. Ceph 对象网关 Throttle Metrics Table

集合名称	指标名称	位字段值	简短描述
throttle-*	**val**	10	目前可用节流
	max	10	最大值 throttle
	get	10	Gets
	get_sum	10	获取数据
	get_or_fail_fail	10	在 get_or_fail 时被阻断
	get_or_fail_success	10	在 get_or_fail 期间获得成功
	take	10	Takes
	take_sum	10	获取的数据
	put	10	Puts
	put_sum	10	放置数据
	wait	5	等待延迟

第 10 章 MCLOCK OSD 调度程序

作为存储管理员，您可以使用 mClock 排队调度程序实施 Red Hat Ceph Storage 的服务质量(QoS)。这基于对名为 dmClock 的 mClock 算法的改编。

mClock OSD 调度程序使用配置文件提供所需的 QoS，以分配正确的保留、权重和将标签限制为服务类型。

mClock OSD 调度程序为不同的设备类型（即 SSD 或 HDD）执行 QoS 计算，方法是在 *mclock 配置选项* 部分中使用 OSD 的 IOPS 功能（自动决定）和最大后续带宽功能(See osd_mclock_max_sequential_bandwidth_hdd 和 osd_mclock_max_sequential_bandwidth_ssd)。

10.1. MCLOCK OSD 调度程序与 WPQ OSD 调度程序的比较

mClock OSD 调度程序是默认调度程序，在旧的 Red Hat Ceph Storage 系统中替换以前的 Weighted Priority Queue (WPQ) OSD 调度程序。

重要

BlueStore OSD 支持 mClock 调度程序。

mClock OSD 调度程序目前具有即时队列，需要立即响应的操作排队。即时队列不由 mClock 处理，它实际上是一个先进先出的队列并有第一优先级。

OSD 复制操作、OSD 操作回复、对等、恢复标记为最高优先级等操作被放入直接队列中。所有其他操作都排队到 mClock 队列中，该队列根据 mClock 算法工作。

mClock 队列 mclock_scheduler 根据它们所属的存储桶进行优先排序，即 pg recovery, pg scrub, snap trim, client op, 和 pg deletion。

在进行后台操作时，与 WPQ 调度程序相比，平均客户端吞吐量(IOPS)的输入和输出操作要高得多，延迟会降低 mClock 配置集。这是因为 mClock 的有效分配 QoS 参数。

其它资源

- 如需更多信息，请参阅 *mClock 配置集* 部分。

10.2. 输入和输出资源的分配

本节论述了 QoS 控制如何在内部使用保留、限制和权重分配。用户不会被预期将这些控制设置为 mClock 配置集会自动设置它们。调优这些控制只能使用可用的 mClock 配置集执行。

dmClock 算法分配 Ceph 集群的输入和输出(I/O)资源，以权重为 weight。它实现了最小保留限制和最大限制，以确保服务可以平平地竞争资源。

目前，mclock_scheduler 操作队列将涉及 I/O 资源的 Ceph 服务划分为以下存储桶：

- 客户端 op ：客户端发布的每秒输入和输出操作(IOPS)。
- PG 删除 ：由主 Ceph OSD 发布的 IOPS。
- snap trim ：快照修剪相关的请求。
- PG 恢复 ：与恢复相关的请求。
- PG scrub ：清理相关的请求。

资源使用以下三组标签进行分区，即每种服务的共享由以下三个标签控制：

- 保留
- 限制
- Weight

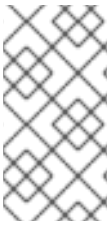
保留

为服务分配的最小 IOPS。服务数量越多，只要服务需要的资源，可以保证这些资源越多。

例如，保留设置为 0.1 （或 10%）的服务始终为自己分配 OSD 的 IOPS 容量的 10%。因此，即使客户端开始发出大量 I/O 请求，它们也不会耗尽所有 I/O 资源，即使具有高负载的集群中服务的操作也不会耗尽。

限制

为服务分配的最大 IOPS。该服务不会超过每秒服务的请求数，即使它要求如此，并且其他服务都不与其竞争。如果服务超过强制限制，则操作队列中会保留在操作队列中，直到恢复限制为止。



注意

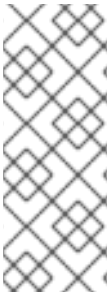
如果值设为 0 （禁用），则服务不受限制设置的限制，如果不存在其他竞争操作，则可以使用所有资源。这在 mClock 配置集中以"MAX"表示。

注意

保留和限制参数分配是每个分片，基于 Ceph OSD 下的后备设备类型，即 HDD 或 SSD。有关 osd_op_num_shards_hdd 和 osd_op_num_shards_ssd 参数的详情，请参阅 *OSD Object storage 守护进程配置选项*。

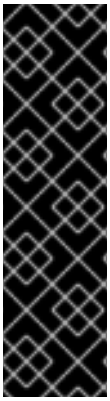
Weight

如果额外容量或系统不够，则容量比例共享。如果权重高于其竞争者的权重，该服务可以使用更大的 I/O 资源。



注意

服务的保留和限制值按 OSD 总 IOPS 容量的比例指定。比例以 mClock 配置集中的百分比表示。权重没有单元。权重相对于另一个请求，因此如果一个类请求的权重为 9，另一个请求权重为 1，则请求在 9 到 1 的比例执行。但是，这只有在满足保留后才会发生，这些值包括在保留阶段执行的操作。



重要

如果 weight 设为 W，那么对于下一个输入的请求的给定类，则输入的权重标签为 1/W，之前的权重标签或当前时间（以较大者为准）。这意味着，如果 W 太大，因此 1/W 太小，则可能无法分配计算的标签，因为它不会获得当前时间的值。

因此，权重的值应始终低于预期每秒可以被处理的请求数量。

10.3. 影响 MCLOCK 操作队列的因素

有三个因素可降低 Red Hat Ceph Storage 中 mClock 操作队列的影响：

- 客户端操作的分片数量。
- 操作序列中的操作数量。
- 为 Ceph OSD 使用分布式系统

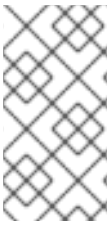
客户端操作的分片数量

对 Ceph OSD 的请求通过其放置组标识符进行分片。每个分片都有自己的 mClock 队列，这些队列彼此不会进行交互，也不会共享信息。

可以使用这些配置选项控制分片数量：

- osd_op_num_shards
- osd_op_num_shards_hdd
- osd_op_num_shards_ssd

较少的分片会增加 mClock 队列的影响，但可能还会有其他影响。



注意

使用由配置选项 osd_op_num_shards、osd_op_num_shards_hdd 和 osd_op_num_shards_ssd 定义的默认分片数量。

操作序列中的操作数量

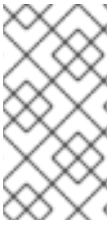
请求从操作队列传输到处理它们的操作序列。mClock 调度程序位于操作队列中。它决定将哪些操作传输到操作序列器。

操作序列中允许的操作数量是一个复杂的问题。目的是在操作序列中保持足够的操作，以便它始终在某些地方正常工作，同时它会等待磁盘和网络访问完成其他操作。

但是，mClock 不再控制传送到操作序列器的操作。因此，为了最大程度提高 mClock 的影响，其目标也是尽可能在操作序列中保留一些操作。

影响操作顺序中操作数量的配置选项有：

- bluestore_throttle_bytes
- bluestore_throttle_deferred_bytes
- bluestore_throttle_cost_per_io
- bluestore_throttle_cost_per_io_hdd
- bluestore_throttle_cost_per_io_ssd



注意

使用由 bluestore_throttle_bytes 和 bluestore_throttle_deferred_bytes 选项定义的默认值。但是，这些选项可以在基准测试阶段确定。

为 Ceph OSD 使用分布式系统

影响 mClock 算法影响的第三个因素是使用分布式系统，其中向多个 Ceph OSD 发出请求，每个 Ceph OSD 可以有多个分片。但是，Red Hat Ceph Storage 目前使用 mClock 算法，它不是 mClock 的分布式版本。

注意

dmClock 是 mClock 的分布式版本。

其它资源

- 有关 osd_op_num_shards_hdd 和 osd_op_num_shards_ssd 参数的详情，请参阅 *Object Storage Daemon (OSD)配置选项*。
- 有关 *BlueStore throttle 参数的详情，请参阅 BlueStore 配置选项*。
- 如需更多信息，请参阅 *手动基准测试 OSD*。

10.4. MCLOCK 配置

mClock 配置集隐藏了用户的低级别详情，以便更轻松地配置和使用 mClock。

mClock 配置集需要以下输入参数来配置服务质量(QoS)相关参数：

- 每个 Ceph OSD 的输入和输出操作的总容量(IOPS)。这是自动确定的。
- 每个操作系统的最大顺序带宽容量(MiB/s)。请参阅 osd_mclock_max_sequential_bandwidth_[hdd/ssd] 选项
- 要启用的 mClock 配置集类型。默认值为 balanced。

利用指定配置文件中的设置，Ceph OSD 会决定并应用较低级别的 mClock 和 Ceph 参数。通过 mClock 配置集应用的参数，可以在 OSD 中的客户端 I/O 和后台操作之间调整 QoS。

其它资源

其它资源

- 有关自动化 *OSD 容量确定的更多信息，请参阅 Ceph* OSD 容量确定。

10.5. MCLOCK 客户端

mClock 调度程序处理来自不同类型的 Ceph 服务的请求。每个服务都由 mClock 视为客户端类型。根据处理的请求类型，mClock 客户端被分类为存储桶：

- client - 处理由 Ceph 外部客户端发布的输入和输出(I/O)请求。
- 后台恢复 - 处理内部恢复请求。
- 背景最佳 - 处理内部回填、清理、snap trim 和放置组(PG)删除请求。

mClock 调度程序从 *osd_mclock_max_capacity_iops_hdd* | *osd_mclock_max_capacity_iops_ssd*, osd_mclock_max_ential_bandwidth_hdd |*osd_mclock_max_sequential_bandwidth_hdd* | *osd_mclock_max_sequential_bandwidth_ssd* 和 *osd_op_num_shards_hdd* | *osd_op_num_shards_ssd* 参数生成在 QoS 计算中使用的操作成本。

10.6. MCLOCK 配置集

mClock 配置集是一个配置设置。当应用到正在运行的 Red Hat Ceph Storage 集群时，它启用了属于不同客户端类的 IOPS 操作节流，如后台恢复、清理、snap trim、client op 和 pg 删除。

mClock 配置集使用用户选择的容量限制和 mClock 配置集类型来确定低级别 mClock 资源控制配置参数，并透明应用它们。另外还会应用其他 Red Hat Ceph Storage 配置参数。低级 mClock 资源控制参数是保留、限制和权重，提供对资源共享的控制。mClock 配置集为每个客户端类型以不同的方式分配这些参数。

10.6.1. mClock 配置集类型

mClock 配置文件可归类为 *内置* 和 *自定义配置集*。

如果有任何 mClock 配置集处于活跃状态，则禁用以下 Red Hat Ceph Storage 配置睡眠选项，这意味着它们被设置为 0 ：

- osd_recovery_sleep
- osd_recovery_sleep_hdd
- osd_recovery_sleep_ssd
- osd_recovery_sleep_hybrid
- osd_scrub_sleep
- osd_delete_sleep
- osd_delete_sleep_hdd
- osd_delete_sleep_ssd
- osd_delete_sleep_hybrid
- osd_snap_trim_sleep
- osd_snap_trim_sleep_hdd
- osd_snap_trim_sleep_ssd
- osd_snap_trim_sleep_hybrid

它是确保 mClock 调度程序能够决定何时从其操作队列中选择下一个操作并将其传送到操作序列器。这会导致在其所有客户端之间提供所需的 QoS。

Custom profile

此配置集允许用户完全控制所有 mClock 配置参数。它应该谨慎使用，对于了解 mClock 和 Red Hat Ceph Storage 相关配置选项的高级用户。

内置 配置集

启用 *内置配置集时*，mClock 调度程序会根据每个客户端类型启用的配置集计算低级别 mClock 参数，即保留、权重和限制。

mClock 参数根据之前提供的最大 Ceph OSD 容量来计算。因此，在使用任何内置配置集时，无法修改以下 mClock 配置选项：

- osd_mclock_scheduler_client_res
- osd_mclock_scheduler_client_wgt
- osd_mclock_scheduler_client_lim
- osd_mclock_scheduler_background_recovery_res
- osd_mclock_scheduler_background_recovery_wgt
- osd_mclock_scheduler_background_recovery_lim
- osd_mclock_scheduler_background_best_effort_res
- osd_mclock_scheduler_background_best_effort_wgt
- osd_mclock_scheduler_background_best_effort_lim

注意

这些默认设置无法使用任何 config 子系统命令（如 config set、config daemon 或 config tell 命令）进行修改。虽然上述命令报告成功，但 mclock QoS 参数会被恢复为对应的内置配置文件默认值。

以下恢复和回填相关的 Ceph 选项会被覆盖为 mClock 默认值：

警告

不要更改这些选项，因为内置配置集会根据它们进行优化。更改这些默认值可能会导致意外的性能结果。

- osd_max_backfills
- osd_recovery_max_active
- osd_recovery_max_active_hdd
- osd_recovery_max_active_ssd

以下选项显示 mClock 默认值，其与当前默认值相同，以最大化前台客户端操作的性能：

osd_max_backfills

原始默认值

1

mClock 默认

1

osd_recovery_max_active

原始默认值

0

mClock 默认

0

osd_recovery_max_active_hdd

原始默认值

3

mClock 默认

3

osd_recovery_max_active_sdd

原始默认值

10

mClock 默认

10

注意

以上 mClock 默认值可以通过启用 osd_mclock_override_recovery_settings （默认设置为 false ）来修改以上 mClock 默认值。请参阅 *修改回填和恢复选项* 来修改这些参数。

***内置* 配置集类型**

用户可以从以下 *内置* 配置集类型中选择：

- **balanced （默认）**

- high_client_ops
- high_recovery_ops

注意

以下列表中提到的值代表为服务类型分配的 Ceph OSD 的总 IOPS 容量。

- balanced ：

默认的 mClock 配置集被设置为 balanced，因为它代表在优先级客户端 IO 或恢复 IO 之间的折衷。它为客户端操作和后台恢复操作分配相等的保留或优先级。后台最佳操作被授予较低的保留速度，因此在竞争操作时需要更长的时间才能完成。此配置集满足集群的正常或稳定状态要求，当外部客户端性能要求不关键时，还有其他后台操作仍需要注意到 OSD 中。

可能存在需要为客户端操作或恢复操作赋予更高优先级的实例。要满足这些要求，您可以选择 high_client_ops 配置集来优先选择客户端 IO 或 high_recovery_ops 配置集来优先恢复 IO。下面将进一步讨论这些配置集。

服务类型：客户端

保留

50%

限制

MAX

Weight

1

服务类型：后台恢复

保留

50%

限制

MAX

Weight

1

服务类型： background best-effort

保留

MIN

限制

90%

Weight

1

- high_client_ops

与 Ceph OSD 中的后台操作相比，此配置集通过分配更多保留和限制来优化后台活动的客户端性能。例如，这个配置集可以启用来为 I/O 密集型应用程序提供所需的性能，以便在恢复较慢的时间内持续运行。以下列表显示了配置集设置的资源控制参数：

服务类型：客户端

保留

60%

限制

MAX

Weight

2

服务类型：后台恢复

保留

40%

限制

MAX

Weight

1

服务类型： background best-effort

保留

MIN

限制

70%

Weight

1

- high_recovery_ops

与外部客户端和其他 Ceph OSD 中的后台操作相比，此配置集会优化后台恢复性能。

例如，管理员可以临时启用它，以便在非高峰期加速后台恢复。以下列表显示了配置集设置的资源控制参数：

服务类型：客户端

保留

30%

限制

MAX

Weight

1

服务类型：后台恢复

保留

70%

限制

MAX

Weight

2

服务类型： background best-effort

保留

MIN

限制

MAX

Weight

1

其它资源

- 有关 *mClock 配置选项的更多信息*，请参阅 mClock 配置选项。

10.6.2. 更改 mClock 配置集

默认的 mClock 配置集设置为 balanced。另一种 *内置* 配置集是 high_client_ops 和 high_recovery_ops。

注意

除非是高级用户，否则不建议 *自定义配置集*。

先决条件

- **一个正在运行的 Red Hat Ceph Storage 集群。**
- **Ceph 监控主机的 root 级别访问权限。**

流程

1. **登录到 Cephadm shell：**

示例

```
[root@host01 ~]# cephadm shell
```

2. **设置 osd_mclock_profile 选项：**

语法

```
ceph config set osd.OSD_ID osd_mclock_profile VALUE
```

示例

```
[ceph: root@host01 /]# ceph config set osd.0 osd_mclock_profile high_recovery_ops
```

这个示例将配置集更改为允许在 **osd.0** 上更快地恢复。

注意

为了获得最佳性能，必须使用以下命令在所有 **Ceph OSD** 上设置配置集：

语法

```
ceph config set osd osd_mclock_profile VALUE
```

10.6.3. 在 *内置* 和 *自定义配置集* 间切换

以下步骤描述了从 *内置* 配置集切换到 *自定义配置集*，反之亦然。

如果要完全控制所有 **mClock** 配置选项，您可能希望切换到 *自定义配置集*。但是，除非您是高级用户，否则建议您不要使用 *自定义配置集*。

先决条件

- 一个正在运行的 **Red Hat Ceph Storage** 集群。
- **Ceph** 监控主机的 **root** 级别访问权限。

从 *内置* 配置集切换到 *自定义配置集*

1. 登录到 **Cephadm shell**：

示例

```
[root@host01 ~]# cephadm shell
```

2. **切换到 *自定义配置集* ：**

语法

```
ceph config set osd.OSD_ID osd_mclock_profile custom
```

示例

```
[ceph: root@host01 /]# ceph config set osd.0 osd_mclock_profile custom
```

注意

为了获得最佳性能，必须使用以下命令在所有 Ceph OSD 上设置配置集：

示例

```
[ceph: root@host01 /]# ceph config set osd osd_mclock_profile custom
```

3. **可选：切换到 *自定义配置集* 后，修改所需的 mClock 配置选项：**

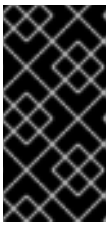
语法

```
ceph config set osd.OSD_ID MCLOCK_CONFIGURATION_OPTION VALUE
```

示例

```
[ceph: root@host01 /]# ceph config set osd.0 osd_mclock_scheduler_client_res 0.5
```

本例将特定 OSD osd.0 的客户端保留 IOPS 比率改为 0.5 (50%)



重要

相应地更改其他服务的保留，如后台恢复和后台最佳状态，以确保保留的总和不超过 OSD 容量的最大比例(1.0)。

从 *自定义配置集* 切换到 *内置* 配置集

1. **登录到 cephadm shell：**

示例

```
[root@host01 ~]# cephadm shell
```

2. **设置所需的 *内置* 配置集：**

语法

```
ceph config set osd osd_mclock_profile MCLOCK_PROFILE
```

示例

```
[ceph: root@host01 /]# ceph config set osd osd_mclock_profile high_client_ops
```

本例在所有 Ceph OSD 上将*内置的*配置集设置为 high_client_ops。

3. **确定数据库中现有的自定义 mClock 配置设置：**

示例

```
[ceph: root@host01 /]# ceph config dump
```

4. **删除之前确定的自定义 mClock 配置设置：**

语法

```
ceph config rm osd MCLOCK_CONFIGURATION_OPTION
```

示例

```
[ceph: root@host01 /]# ceph config rm osd osd_mclock_scheduler_client_res
```

本例删除所有 Ceph OSD 上设置的配置选项 osd_mclock_scheduler_client_res。

从中央配置数据库中删除所有现有的自定义 mClock 配置设置后，将应用与 high_client_ops 相关的配置设置。

5. 验证 Ceph OSD 上的设置：

语法

```
ceph config show osd.OSD_ID
```

示例

```
[ceph: root@host01 /]# ceph config show osd.0
```

其它资源

- 如需无法使用*内置*配置集修改的 mClock 配置选项列表，请参阅 *mClock 配置集类型*。

10.6.4. 在 mClock 配置集间临时切换

本节包含在 mClock 配置文件之间临时切换的步骤。

警告

本节适用于高级用户或实验性测试。不要在正在运行的存储集群中使用以下命令，因为它可能会造成意外的结果。

注意

使用以下命令，Ceph OSD 上的配置更改是临时的，在 Ceph OSD 重启时会丢失。

重要

使用本节中描述的命令覆盖的配置选项无法使用 ceph config set osd.*OSD_ID* 命令进一步修改。在重启给定的 Ceph OSD 前，这些更改才会生效。这是有意设计的，根据配置子系统设计。但是，仍可使用这些命令临时进行任何进一步的修改。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。
- Ceph 监控主机的 root 级别访问权限。

流程

1. 登录到 Cephadm shell：

示例

```
[root@host01 ~]# cephadm shell
```

2. **运行以下命令来覆盖 mClock 设置：**

语法

```
ceph tell osd.OSD_ID injectargs '--MCLOCK_CONFIGURATION_OPTION=VALUE'
```

示例

```
[ceph: root@host01 /]# ceph tell osd.0 injectargs '--osd_mclock_profile=high_recovery_ops'
```

这个示例覆盖 osd.0 上的 osd_mclock_profile 选项。

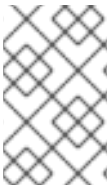
3. **可选： 您可以使用上一个 ceph 的替代方案 tell osd.*OSD_ID* injectargs 命令：**

语法

```
ceph daemon osd.OSD_ID config set MCLOCK_CONFIGURATION_OPTION VALUE
```

示例

```
[ceph: root@host01 /]# ceph daemon osd.0 config set osd_mclock_profile
high_recovery_ops
```



注意

也可以使用上述命令临时修改 *自定义配置集* 的单独 QoS 相关配置选项。

10.6.5. 使用 mClock 配置集降级和未替换的对象恢复率

降级对象恢复被归类到后台恢复存储桶中。在所有 mClock 配置集中，与 misplaced 对象恢复相比，降级对象会被赋予更高的优先级，因为降级对象会出现一个数据安全问题，而这些对象不会被错误覆盖。

回填或错误对象恢复操作被归类到后台 best-effort bucket 中。根据 balanced 和 high_client_ops mClock 配置集，后台 best-effort 客户端不会被保留（设置为零）限制，但如果不存在其他竞争服务，则限制使用参与 OSD 的容量。

因此，使用 balanced 或 high_client_ops 配置集，与前面的 *WeightedPriorityQueue* (WPQ)调度程序相比，回填率应该会较慢。

如果需要更高的回填率，请按照以下部分所述的步骤进行操作。

提高回填率

要使用 balanced 或 high_client_ops 配置集时更快地回填率，请按照以下步骤执行：

- 在回填期间切换到 'high_recovery_ops' mClock 配置集。请参阅 *更改 mClock 配置集* 来实现此目的。回填阶段完成后，将 mClock 配置集切换到之前活跃的配置集。如果带有 'high_recovery_ops' 配置集的回填率没有显著改进，请继续下一步。
- 将 mClock 配置集切回到之前活跃的配置集。
- 将 'osd_max_backfills' 修改为更高的值，例如 3。请参阅 *修改回填和恢复选项* 来实现此目的。
- 回填完成后，"osd_max_backfills"可以按照第 3 步所述的相同步骤重置为默认值 1。

警告

请注意，修改 osd_max_backfills 可能会导致其他操作，例如，客户端操作在回填阶段可能会遇到更高的延迟。因此，建议用户以较小的增量增加 osd_max_backfills，以最小化对集群中其他操作的性能影响。

10.6.6. 修改 回填和恢复 选项

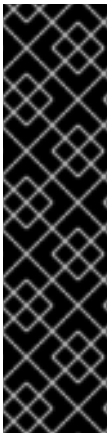
使用 ceph config set 命令修改 backfills 和 recovery 选项。

可以修改的回填或恢复选项列在 *mClock 配置集类型* 中。

警告

本节适用于高级用户或实验性测试。不要在正在运行的存储集群中使用以下命令，因为它可能会造成意外的结果。

仅修改实验性测试的值，或者集群无法处理这些值，或使用默认设置显示性能不佳。



重要

对 mClock 默认回填或恢复选项的修改由 osd_mclock_override_recovery_settings 选项限制，该选项默认设置为 false。

如果您试图修改任何默认的回填或恢复选项，而不将 osd_mclock_override_recovery_settings 设置为 true，它会将选项重置为 mClock 默认值，以及集群日志记录中记录的警告信息。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。
- Ceph 监控主机的 root 级别访问权限。

流程

1. 登录到 Cephadm shell：

示例

```
[root@host01 ~]# cephadm shell
```

2. 在所有 Ceph OSD 上，将 osd_mclock_override_recovery_settings 配置选项设为 true：

示例

```
[ceph: root@host01 /]# ceph config set osd osd_mclock_override_recovery_settings true
```

3. 设置所需的 回填 或 恢复选项：

语法

```
ceph config set osd OPTION VALUE
```

示例

```
[ceph: root@host01 /]# ceph config set osd osd_max_backfills_ 5
```

4. **等待几秒钟，并验证特定 OSD 的配置：**

语法

```
ceph config show osd.OSD_ID_ | grep OPTION
```

示例

```
[ceph: root@host01 /]# ceph config show osd.0 | grep osd_max_backfills
```

5. **在所有 OSD 上，将 osd_mclock_override_recovery_settings 配置选项重置为 false ：**

示例

```
[ceph: root@host01 /]# ceph config set osd osd_mclock_override_recovery_settings false
```

10.7. CEPH OSD 容量确定

总 IOPS 的 Ceph OSD 容量在 Ceph OSD 初始化期间自动决定。这可以通过运行 Ceph OSD bench 工具，并覆盖 osd_mclock_max_capacity_iops_[hdd, ssd] 选项的默认值，具体取决于设备类型。用户不需要其他操作或输入来设置 Ceph OSD 容量。

从自动化过程缓解非实际 Ceph OSD 容量

在某些情况下，Ceph OSD bench 工具可能会根据驱动器配置和其他环境相关条件显示不切的或有问题的结果。

要降低由于这种不切容量造成的性能影响，定义了几个阈值配置选项，具体取决于 OSD 设备类型并使用：

- osd_mclock_iops_capacity_threshold_hdd = 500
- osd_mclock_iops_capacity_threshold_ssd = 80000

您可以运行以下命令来验证这些参数：

```
[ceph: root@host01 /]# ceph config show osd.0 osd_mclock_iops_capacity_threshold_hdd
500.000000
[ceph: root@host01 /]# ceph config show osd.0 osd_mclock_iops_capacity_threshold_ssd
80000.000000
```

注意

如果要手动基准测试 OSD 或手动调整 BlueStore throttle 参数，请参阅 *手动基准测试 OSD*。

您可以运行以下命令来在集群启动后验证 OSD 容量：

语法

```
ceph config show osd.N osd_mclock_max_capacity_iops_[hdd, ssd]
```

示例

```
[ceph: root@host01 /]# ceph config show osd.0 osd_mclock_max_capacity_iops_ssd
```

在上例中，您可以查看 Red Hat Ceph Storage 节点上的 osd.0 的最大容量，其底层设备是 SSD。

执行以下自动步骤：

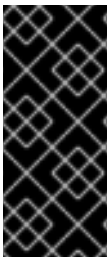
回退到使用默认 OSD 容量

如果 Ceph OSD bench 工具报告超过上述阈值的测量，则回退机制将恢复到 osd_mclock_max_capacity_iops_hdd 或 osd_mclock_max_capacity_iops_ssd 的默认值。阈值配置选项可以根据所使用的驱动器类型重新配置。

当测量超过阈值时，会记录集群警告：

示例

```
3403 Sep 11 11:52:50 dell-r640-039.dsal.lab.eng.rdu2.redhat.com ceph-osd[70342]:
log_channel(cluster) log [WRN] : OSD bench result of 49691.213005 IOPS exceeded the threshold
limit of 500.000000 IOPS for osd.27. IOPS capacity is unchanged at 315.000000 IOPS. The
recommendation is to establish the osd's IOPS capacity using other benchmark tools (e.g. Fio) and
then override osd_mclock_max_capacity_iops_[hdd|ssd].
```



重要

如果默认容量无法准确代表 Ceph OSD 容量，则强烈建议您使用首选工具运行自定义基准，如驱动器上的 Fio，然后覆盖 osd_mclock_max_capacity_iops_[hdd, ssd] 选项，如 *指定最大 OSD 容量* 中所述。

其它资源

- 请参阅 *手动基准测试 OSD* 以手动基准测试 Ceph OSD 或手动调整 BlueStore throttle 参数。
- 有关 osd_mclock_max_capacity_iops_[hdd, ssd] 和 osd_mclock_iops_capacity_threshold_[hdd, ssd] 选项的更多信息，请参阅 *mClock 配置选项*。

10.7.1. 验证 OSD 的容量

您可以在设置存储集群后验证 Ceph OSD 的容量。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。
- Ceph 监控主机的 root 级别访问权限。

流程

1. 登录到 Cephadm shell：

   示例

   ```
   [root@host01 ~]# cephadm shell
   ```

2. 验证 Ceph OSD 的容量：

   语法

   ```
   ceph config show osd.OSD_ID osd_mclock_max_capacity_iops_[hdd, ssd]
   ```

-

示例

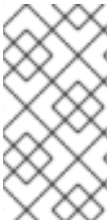
```
[ceph: root@host01 /]# ceph config show osd.0 osd_mclock_max_capacity_iops_ssd

21500.000000
```

10.7.2. 手动基准测试 OSD

要手动对 Ceph OSD 进行基准测试，可以使用任何现有的基准测试工具，如 Fio。无论所使用的工具或命令是什么，以下步骤仍然保持不变。

重要

分片和 BlueStore throttle 参数的数量会影响 mClock 操作队列。因此，务必要仔细设置这些值，以便最大程度提高 mclock 调度程序的影响。有关这些值的更多信息，请参阅 *影响 mClock 操作队列的因素*。



注意

只有在您要覆盖 OSD 初始化期间自动确定的 Ceph OSD 容量时，才需要执行本节中的步骤。

注意

如果您已经确定了基准数据并希望手动覆盖 Ceph OSD 的最大 OSD 容量，请跳至 *指定最大 OSD 容量* 部分。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。

- **Ceph 监控主机的 root 级别访问权限。**

流程

1. **登录到 Cephadm shell：**

   示例

   ```
   [root@host01 ~]# cephadm shell
   ```

2. **Ceph OSD 基准：**

   语法

   ```
   ceph tell osd.OSD_ID bench [TOTAL_BYTES] [BYTES_PER_WRITE] [OBJ_SIZE]
   [NUM_OBJS]
   ```

   其中：

   - ***TOTAL_BYTES*: 要写入的字节数。**
   - ***BYTES_PER_WRITE* ：每个写入的块大小。**
   - ***OBJ_SIZE*: 每个对象字节.**
   - ***NUM_OBJS* ：要写入的对象数。**

示例

```
[ceph: root@host01 /]# ceph tell osd.0 bench 12288000 4096 4194304 100
{
    "bytes_written": 12288000,
    "blocksize": 4096,
    "elapsed_sec": 1.3718913019999999,
    "bytes_per_sec": 8956977.8466311768,
    "iops": 2186.7621695876896
}
```

10.7.3. 确定正确的 BlueStore 节流值

此可选部分详细介绍了用于确定正确的 BlueStore throttle 值的步骤。这些步骤使用默认的分片。

重要

在运行测试前，清除缓存以获取准确的测量。使用以下命令清除每个基准之间的 OSD 缓存：

语法

```
ceph tell osd.OSD_ID cache drop
```

示例

```
[ceph: root@host01 /]# ceph tell osd.0 cache drop
```

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。
- 对托管您要基准测试的 OSD 的 Ceph 监控节点的根级别访问权限。

流程

1. 登录到 Cephadm shell：

示例

```
[root@host01 ~]# cephadm shell
```

2. 在 OSD 上运行简单的 4KiB 随机写入工作负载：

语法

```
ceph tell osd.OSD_ID bench 12288000 4096 4194304 100
```

示例

```
[ceph: root@host01 /]# ceph tell osd.0 bench 12288000 4096 4194304 100
{
    "bytes_written": 12288000,
    "blocksize": 4096,
    "elapsed_sec": 1.3718913019999999,
    "bytes_per_sec": 8956977.8466311768,
    "iops": 2186.7621695876896 1
}
```

1

从 osd bench 命令的输出中获取的整体吞吐量。这个值是基准吞吐量，当默认的 BlueStore throttle 选项生效时。

3. 请注意，总体吞吐量（即 IOPS ）从上一命令的输出中获取。

4. 如果目的是确定环境的 BlueStore 节流值，请将 bluestore_throttle_bytes 和 bluestore_throttle_deferred_bytes 选项设置为 32 KiB，即 32768 Bytes：

语法

```
ceph config set osd.OSD_ID bluestore_throttle_bytes 32768
ceph config set osd.OSD_ID bluestore_throttle_deferred_bytes 32768
```

示例

```
[ceph: root@host01 /]# ceph config set osd.0 bluestore_throttle_bytes 32768
[ceph: root@host01 /]# ceph config set osd.0 bluestore_throttle_deferred_bytes 32768
```

否则，您可以跳至下一部分 *指定最大 OSD 容量*。

5. 使用 OSD bench 命令前，先运行 4KiB 随机写入测试：

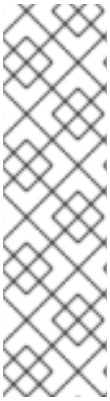
示例

```
[ceph: root@host01 /]# ceph tell osd.0 bench 12288000 4096 4194304 100
```

6. 注意输出的整体吞吐量，并将值与之前记录的基准吞吐量进行比较。

7. 如果吞吐量与基准不匹配，请通过将 BlueStore 节流选项乘以 2 来提高 BlueStore 节流选项。

8. 运行 4KiB 随机写入测试，将值与基准吞吐量进行比较，并将 BlueStore 节流选项乘以 2，直到获取的吞吐量非常接近基准值。



注意

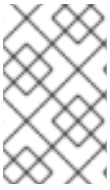
例如，在对具有 NVMe SSD 的机器进行基准测试期间，BlueStore 节流和延迟字节数都有一个 256 KiB 值，这决定了最大化 mClock 的影响。对于 HDD，对应的值为 40 MiB，其中整个吞吐量大致等于基准吞吐量。

对于 HDD，与 SSD 相比，BlueStore throttle 值应该较高。

10.7.4. 指定最大 OSD 容量

您可以覆盖 OSD 初始化过程中自动设置的最大 Ceph OSD 容量。

这些步骤是可选的。如果默认容量无法准确代表 Ceph OSD 容量，请执行以下步骤。



注意

确保您首先确定基准数据，如 *手动基准测试 OSD* 中所述。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。

- **Ceph 监控主机的 root 级别访问权限。**

流程

1. **登录到 Cephadm shell：**

   示例

   ```
   [root@host01 ~]# cephadm shell
   ```

2. **为 OSD 设置 osd_mclock_max_capacity_iops_[hdd, ssd] 选项：**

   语法

   ```
   ceph config set osd.OSD_ID osd_mclock_max_capacity_iops_[hdd,ssd] VALUE
   ```

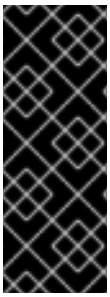
   示例

   ```
   [ceph: root@host01 /]# ceph config set osd.0 osd_mclock_max_capacity_iops_hdd 350
   ```

   本例将 osd.0（其底层设备类型为 HDD）的最大容量设置为 350 IOPS。

第 11 章 BLUESTORE

BlueStore 是 OSD 守护进程的后端对象存储，直接在块设备上放置对象。



重要

BlueStore 为生产环境中的 OSD 守护进程提供了高性能后端。默认情况下，BlueStore 配置为自我调整。如果您确定您的环境使用手动调优的 BlueStore 的性能更好，请联系红帽支持并共享您的配置详情，以帮助我们改进自动调整功能。红帽期待您的反馈意见并感谢您的建议。

11.1. CEPH BLUESTORE

以下是使用 BlueStore 的一些主要功能：

直接管理存储设备

BlueStore 使用原始块设备或分区。这可避免任何抽象层的抽象层，如 XFS 等本地文件系统，这可能会限制性能或增加复杂性。

使用 RocksDB 进行元数据管理

BlueStore 使用 RocksDB key-value 数据库来管理内部元数据，如从对象名称到磁盘上的块位置的映射。

完整数据和元数据校验和

默认情况下，写入 BlueStore 的所有数据和元数据都受到一个或多个校验和的保护。在不验证的情况下，不会从磁盘或返回给用户读取数据或元数据。

内联压缩

在写入磁盘前，可以选择性地压缩数据。

高效的 copy-on-write 功能

Ceph 块设备和 Ceph 文件系统快照依赖于在 BlueStore 中高效实施的写时复制克隆机制。这可以提高常规快照以及依赖克隆来实现高效的两阶段提交的纠删代码池的 I/O 效率。

没有大型的双写方式

BlueStore 首先将任何新数据写入块设备上未分配空间，然后提交 RocksDB 事务来更新对象元数据以引用磁盘的新区域。只有在写入操作低于可配置的大小阈值时，它会返回 write-ahead 日志方案。

支持多设备

BlueStore 可以使用多个块设备来存储不同的数据。例如：用于数据的硬盘驱动器(HDD)，用于元数据的 Solid-state Drive(SSD)，即 Non-volatile Memory(NVM)或 Non-volatile random-access memory(NVRAM)或 RocksDB write-ahead 日志的持久性内存。详情请参阅 *Ceph BlueStore 设备*。

高效的块设备使用

因为 BlueStore 不使用任何文件系统，所以它将尽可能减少需要清除存储设备缓存的需要。

分配元数据

分配元数据不再使用 RocksDB 中的独立对象，因为分配信息可以从存储在 RocksDB 中的系统中的所有 onodes 的聚合分配状态中推断出来。BlueStore V3 代码会在分配时间跳过 RocksDB 更新，并在 umount 期间执行分配器对象以及所有 OSD 分配状态的完整取消暂存。这会使 IOPS 增加 25%，并在小的随机写工作负载中降低延迟；但是，它延长了恢复时间，通常要花费几分钟时间，并在失败的情况下不会调用 umount，因为您需要迭代所有 onodes 来重新创建分配元数据。

缓存期限组合

Red Hat Ceph Storage 将不同缓存中的项目与"age bins"相关联，它提供了所有缓存项目的相对年龄视图。

11.2. CEPH BLUESTORE 设备

BlueStore 管理后端中的一个、两个或三个存储设备。

- 主
- WAL
- DB

在最简单的情形中，BlueStore 使用一个主存储设备。存储设备通常作为一个整体使用，占据由 BlueStore 直接管理的完整设备。主设备由数据目录中的 块 符号链接识别。

数据目录是一个 tmpfs 挂载，它填充保存 OSD 信息的所有通用 OSD 文件，如标识符、它所属的集群及其专用密钥环。

存储设备分为两个部分，其中包括：

- OSD 元数据 ：使用 XFS 格式化的小分区，其中包含 OSD 的基本元数据。此数据目录包含 OSD 的信息，如其标识符、所属集群及其专用密钥环。

- 数据 ：一个大型分区，占据由 BlueStore 直接管理的设备的其余部分，其中包含所有 OSD 数据。这个主要设备由数据目录中的块符号链接识别。

您还可以使用两个附加设备：

- WAL(write-ahead-log)设备 ：存储 BlueStore 内部日志或 write-ahead 日志的设备。它通过数据目录中的 block.wal 符号链接来识别。只有在设备比主设备更快时才请考虑使用 WAL 设备。例如，当 WAL 设备使用 SSD 磁盘且主设备使用 HDD 磁盘时。

- DB 设备 ：存储 BlueStore 内部元数据的设备。嵌入式 RocksDB 数据库包含尽可能多的元数据，因为它可以在 DB 设备上而不是主设备上的元数据来提高性能。如果 DB 设备已满，它开始向主设备添加元数据。只有在设备比主设备更快时才考虑使用 DB 设备。

警告

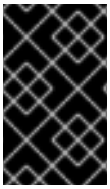
如果您在快速设备中只有 1GB 的存储可用，红帽建议将其用作 WAL 设备。如果您使用更快速的设备可用，请考虑将其用作 DB 设备。BlueStore 日志始终放在最快的设备上，因此使用 DB 设备提供相同的好处，同时允许存储额外的元数据。

11.3. CEPH BLUESTORE 缓存

BlueStore 缓存是缓冲区的集合，具体取决于配置，可以填充数据，因为 OSD 守护进程从中读取或写入到磁盘。默认情况下，Red Hat Ceph Storage 中 BlueStore 将缓存读内容，但不进行写入。这是因为 bluestore_default_buffered_write 选项设置为 false，以避免出现与缓存驱除相关的潜在开销。

如果 bluestore_default_buffered_write 选项被设置为 true，则数据会首先写入缓冲区，然后提交到磁盘。之后，写入确认将发送到客户端，从而加快对缓存中已存在的数据的后续读取速度，直到数据被驱除为止。

对于有大量读操作的工作负载不会立即从 BlueStore 缓存中受益。完成更多阅读后，缓存会随时间增长，后续的读取将提高性能。缓存填充的速度取决于 BlueStore 块和数据库磁盘类型，以及客户端的工作负载要求。



重要

在启用 bluestore_default_buffered_write 选项前，请联系红帽支持。

缓存期限组合

Red Hat Ceph Storage 将不同缓存中的项目与"age bins"相关联，它提供了所有缓存项目的相对年龄视图。例如，当 BlueStore onode 缓存中存在旧的 onode 条目时，会针对单个大型对象发生热读取工作负载。该 OSD 的优先级缓存将旧的 onode 条目排序为比 hot 对象的缓冲缓存数据低的优先级级别。虽然通常，在给定优先级级别上，Ceph 可能会大量优先选择 onodes，但在这种热工作负载场景中，可能会为旧的 onodes 分配比热工作负载数据较低的优先级级别，以便首先实现缓冲区数据内存请求。

11.4. CEPH BLUESTORE 的大小考虑

使用 BlueStore OSD 混合传统和固态硬盘时，必须相应地调整 RocksDB 逻辑卷(block.db)。红帽建议，RocksDB 逻辑卷不要小于使用对象、文件和混合工作负载的块大小的 4%。红帽通过 RocksDB 和 OpenStack 块工作负载支持 1% 的 BlueStore 块大小。例如，如果对象工作负载的块大小为 1 TB，则至少创建 40 GB RocksDB 逻辑卷。

如果没有混合驱动器类型，则不需要单独的 RocksDB 逻辑卷。BlueStore 将自动管理 RocksDB 的大小。

BlueStore 的缓存内存用于 RocksDB、BlueStore 元数据和对象数据的键值对元数据。

注意

除了 OSD 已消耗的内存占用空间外，BlueStore 缓存内存值也除外。

11.5. 使用 BLUESTORE_MIN_ALLOC_SIZE 参数调优 CEPH BLUESTORE

此过程适用于新的或全新部署的 OSD。

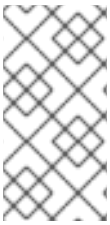
在 BlueStore 中，原始分区在 bluestore_min_alloc_size 的块中分配和管理。默认情况

下，bluestore_min_alloc_size 为 4096，相当于 4 KiB 用于 HDD 和 SSD。当每个块中的未写入区域写入原始分区时，会用零填充。当您没有针对的工作负载正确配置大小时，这可能会导致浪费掉未使用的空间（例如，当编写小对象时）。

最佳实践是将 bluestore_min_alloc_size 设置为与最小写入匹配，以便避免这种写入放大损失。

重要

不建议更改 bluestore_min_alloc_size 的值。如需任何帮助，请联系*红帽支持*。



注意

设置 bluestore_min_alloc_size_ssd 和 bluestore_min_alloc_size_hdd 分别特定于 SSD 和 HDD，但设置它们是必需的，因为设置 bluestore_min_alloc_size 覆盖它们。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。
- Ceph 监视器和管理器已在集群中部署。
- 可以重新置备为 OSD 节点的服务器或节点
- 如果您要重新部署现有的 Ceph OSD 节点，Ceph 监控节点的管理密钥环。

流程

1. 在 bootstrapped 节点上，更改 bluestore_min_alloc_size 参数的值：

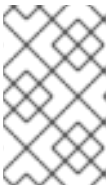
语法

```
ceph config set osd.OSD_ID bluestore_min_alloc_size_DEVICE_NAME_ VALUE
```

示例

```
[ceph: root@host01 /]# ceph config set osd.4 bluestore_min_alloc_size_hdd 8192
```

您可以看到 bluestore_min_alloc_size 设置为 8192 字节，相当于 8 KiB。



注意

所选值应该与 2 的指数相对应。

2. **重新启动 OSD 的服务。**

语法

```
systemctl restart SERVICE_ID
```

示例

```
[ceph: root@host01 /]# systemctl restart ceph-499829b4-832f-11eb-8d6d-
001a4a000635@osd.4.service
```

验证

- 使用 ceph daemon 命令验证设置：

语法

```
ceph daemon osd.OSD_ID config get bluestore_min_alloc_size__DEVICE_
```

示例

```
[ceph: root@host01 /]# ceph daemon osd.4 config get bluestore_min_alloc_size_hdd

ceph daemon osd.4 config get bluestore_min_alloc_size
{
    "bluestore_min_alloc_size": "8192"
}
```

其它资源

- 对于 OSD 移除和添加，请参阅 *Red Hat Ceph Storage Operations Guide* 中的*使用 Ceph Orchestrator 管理 OSD*一章以及以下链接。对于已部署的 OSD，您将无法修改 bluestore_min_alloc_size 参数，因此您必须移除 OSD 并再次部署它们。

11.6. 使用 BLUESTORE 管理工具重新划分 ROCKSDB 数据库

您可以使用 BlueStore 管理工具重新划分数据库。它将 BlueStore 的 RocksDB 数据库从一形转换为几列的系列，而无需重新部署 OSD。列系列的功能与整个数据库相同，但允许用户在较小的数据集上运行并应用不同的选项。它利用存储的键的不同生命周期。密钥会在转换过程中移动，而不会创建新密钥或删除现有密钥。

重新定义 OSD 的方法有两种：

1. 使用 rocksdb-resharding.yml playbook。

2. 手动重新划分 OSD。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。
- 对象存储配置为 BlueStore。
- 部署在主机上的 OSD 节点。
- 对所有主机的 root 级别访问。
- 在所有主机上安装 ceph-common 和 cephadm 软件包。

11.6.1. 使用 rocksdb-resharding.yml playbook

1. 在管理节点上，以 root 用户身份导航到安装 playbook 的 cephadm 文件夹：

示例

```
[root@host01 ~]# cd /usr/share/cephadm-ansible
```

2. 运行 playbook：

语法

```
ansible-playbook -i hosts rocksdb-resharding.yml -e osd_id=OSD_ID -e
admin_node=HOST_NAME
```

示例

```
[root@host01 ~]# ansible-playbook -i hosts rocksdb-resharding.yml -e osd_id=7 -e
admin_node=host03

.............
TASK [stop the osd]
**********************************************************************************************************
**************************************************************************
Wednesday 29 November 2023  11:25:18 +0000 (0:00:00.037)       0:00:03.864 ****
changed: [localhost -> host03]
TASK [set_fact ceph_cmd]
**********************************************************************************************************
**************************************************************************
Wednesday 29 November 2023  11:25:32 +0000 (0:00:14.128)       0:00:17.992 ****
ok: [localhost -> host03]

TASK [check fs consistency with fsck before resharding]
**********************************************************************************************************
****************************************
Wednesday 29 November 2023  11:25:32 +0000 (0:00:00.041)       0:00:18.034 ****
ok: [localhost -> host03]

TASK [show current sharding]
**********************************************************************************************************
**********************************************************************
Wednesday 29 November 2023  11:25:43 +0000 (0:00:11.053)       0:00:29.088 ****
ok: [localhost -> host03]

TASK [reshard]
**********************************************************************************************************
**************************************************************************************
Wednesday 29 November 2023  11:25:45 +0000 (0:00:01.446)       0:00:30.534 ****
ok: [localhost -> host03]

TASK [check fs consistency with fsck after resharding]
**********************************************************************************************************
****************************************
Wednesday 29 November 2023  11:25:46 +0000 (0:00:01.479)       0:00:32.014 ****
ok: [localhost -> host03]

TASK [restart the osd]
**********************************************************************************************************
**************************************************************************
Wednesday 29 November 2023  11:25:57 +0000 (0:00:10.699)       0:00:42.714 ****
changed: [localhost -> host03]
```

3. 验证重新划分已经完成。

a. 停止重新划分的 OSD：

示例

```
[ceph: root@host01 /]# ceph orch daemon stop osd.7
```

b. 进入 OSD 容器：

示例

```
[root@host03 ~]# cephadm shell --name osd.7
```

c. 检查重新划分：

示例

```
[ceph: root@host03 /]# ceph-bluestore-tool --path /var/lib/ceph/osd/ceph-7/ show-sharding
    m(3) p(3,0-12) O(3,0-13) L P
```

.

d. 启动 **OSD：**

示例

```
[ceph: root@host01 /]# ceph orch daemon start osd.7
```

11.6.2. 手动重新划分 OSD

1. 登录 **cephadm shell：**

示例

```
[root@host01 ~]# cephadm shell
```

2. **从管理节点获取 *OSD_ID* 和主机详情：**

示例

```
[ceph: root@host01 /]# ceph orch ps
```

3. **以 root 用户身份登录对应的主机，再停止 OSD：**

语法

```
cephadm unit --name OSD_ID stop
```

示例

```
[root@host02 ~]# cephadm unit --name osd.0 stop
```

4. **进入已停止的 OSD 守护进程容器：**

语法

```
cephadm shell --name OSD_ID
```

示例

```
[root@host02 ~]# cephadm shell --name osd.0
```

5. **登录到 cephadm shell 并检查文件系统一致性：**

语法

```
ceph-bluestore-tool --path/var/lib/ceph/osd/ceph-OSD_ID/ fsck
```

示例

```
[ceph: root@host02 /]# ceph-bluestore-tool --path /var/lib/ceph/osd/ceph-0/ fsck

fsck success
```

6. 检查 **OSD** 节点的分片状态：

语法

```
ceph-bluestore-tool --path /var/lib/ceph/osd/ceph-OSD_ID/ show-sharding
```

示例

```
[ceph: root@host02 /]# ceph-bluestore-tool --path /var/lib/ceph/osd/ceph-6/ show-sharding

m(3) p(3,0-12) O(3,0-13) L P
```

7. 运行 **ceph-bluestore-tool** 命令来重新划分。红帽建议使用命令中的参数：

语法

```
ceph-bluestore-tool --log-level 10 -l log.txt --path /var/lib/ceph/osd/ceph-OSD_ID/ --
sharding="m(3) p(3,0-12) O(3,0-13)=block_cache={type=binned_lru} L P" reshard
```

示例

```
[ceph: root@host02 /]# ceph-bluestore-tool --path /var/lib/ceph/osd/ceph-6/ --sharding="m(3)
p(3,0-12) O(3,0-13)=block_cache={type=binned_lru} L P" reshard

reshard success
```

8. **要检查 OSD 节点的分片状态，请运行 show-sharding 命令：**

语法

```
ceph-bluestore-tool --path /var/lib/ceph/osd/ceph-OSD_ID/ show-sharding
```

示例

```
[ceph: root@host02 /]# ceph-bluestore-tool --path /var/lib/ceph/osd/ceph-6/ show-sharding

m(3) p(3,0-12) O(3,0-13)=block_cache={type=binned_lru} L P
```

9. **从 cephadm shell 退出：**

```
[ceph: root@host02 /]# exit
```

10. **以 root 用户身份登录对应的主机，再启动 OSD：**

语法

```
cephadm unit --name OSD_ID start
```

示例

```
[root@host02 ~]# cephadm unit --name osd.0 start
```

其它资源

- 如需更多信息，请参阅 *Red Hat Ceph Storage 安装指南*。

11.7. BLUESTORE 碎片工具

作为存储管理员，您要定期检查 BlueStore OSD 的碎片级别。您可以通过一个简单命令检查碎片级别，以便离线或在线 OSD。

11.7.1. BlueStore 碎片工具是什么？

对于 BlueStore OSD，可用空间随时间在底层存储设备上发生碎片。有些碎片是正常的，但在出现过量碎片时会导致性能降低。

BlueStore 碎片工具在 BlueStore OSD 的碎片级别生成分数。此碎片分数指定为范围 0 到 1。分数为 0 表示没有碎片，1 分代表严重碎片。

表 11.1. 碎片分数的含义

分数	碎片挂载
0.0 - 0.4	没有或非常少的碎片。

分数	碎片挂载
0.4 - 0.7	较少且可接受的碎片。
0.7 - 0.9	需要考虑但安全的碎片。
0.9 - 1.0	严重碎片会导致性能问题。

重要

如果您存在严重碎片，且需要一些有助于解决问题，请联络红帽支持。

11.7.2. 检查碎片

可以在线或离线检查 BlueStore OSD 的碎片级别。

先决条件

- **一个正在运行的 Red Hat Ceph Storage 集群。**
- **BlueStore OSDs.**

在线 BlueStore 碎片得分

1. **检查正在运行的 BlueStore OSD 进程：**

   a. **简单报告：**

   语法

   ```
   ceph daemon OSD_ID bluestore allocator score block
   ```

   示例

```
[ceph: root@host01 /]# ceph daemon osd.123 bluestore allocator score block
```

b. **详细的报告：**

语法

```
ceph daemon OSD_ID bluestore allocator dump block
```

示例

```
[ceph: root@host01 /]# ceph daemon osd.123 bluestore allocator dump block
```

离线 BlueStore 碎片得分

1. **重新定义 BlueStore OSD。**

语法

```
[root@host01 ~]# cephadm shell --name osd.ID
```

示例

```
[root@host01 ~]# cephadm shell --name osd.2
Inferring fsid 110bad0a-bc57-11ee-8138-fa163eb9ffc2
Inferring config /var/lib/ceph/110bad0a-bc57-11ee-8138-fa163eb9ffc2/osd.2/config
Using ceph image with id `17334f841482` and tag `ceph-7-rhel-9-containers-candidate-59483-20240301201929` created on 2024-03-01 20:22:41 +0000 UTC
registry-proxy.engineering.redhat.com/rh-osbs/rhceph@sha256:09fc3e5baf198614d70669a106eb87dbebee16d4e91484375778d4adbccadacd
```

2. **检查非运行 BlueStore OSD 进程。**

   a. **对于简单报告，请运行以下命令：**

   语法

   ```
   ceph-bluestore-tool --path PATH_TO_OSD_DATA_DIRECTORY --allocator block free-score
   ```

   示例

   ```
   [root@host01 /]# ceph-bluestore-tool --path /var/lib/ceph/osd/ceph-123 --allocator block free-score
   ```

   b. **如需更详细的报告，请运行以下命令：**

   语法

   ```
   ceph-bluestore-tool --path PATH_TO_OSD_DATA_DIRECTORY --allocator block free-dump
   ```

```
block:
{
    "fragmentation_rating": 0.018290238194701977
}
```

示例

```
[root@host01 /]# ceph-bluestore-tool --path /var/lib/ceph/osd/ceph-123 --allocator block
free-dump
block:
{
    "capacity": 21470642176,
    "alloc_unit": 4096,
    "alloc_type": "hybrid",
    "alloc_name": "block",
    "extents": [
        {
            "offset": "0x370000",
            "length": "0x20000"
        },
        {
            "offset": "0x3a0000",
            "length": "0x10000"
        },
        {
            "offset": "0x3f0000",
            "length": "0x20000"
        },
        {
            "offset": "0x460000",
            "length": "0x10000"
        },
```

其它资源

- 有关 *碎片分数的详细信息，请参阅 BlueStore Fragation Tool*。
- 如需了解有关重新划分的详细信息，*请参阅使用 BlueStore 管理工具重新划分 RocskDB 数据库*。

11.8. CEPH BLUESTORE BLUEFS

BlueStore 块数据库将元数据作为键值对存储在 RocksDB 数据库中。块数据库驻留在存储设备上的一个小型 BlueFS 分区。BlueFS 是一个最小的文件系统，旨在保存 RocksDB 文件。

BlueFS 文件

以下是 RocksDB 生成的三种文件类型：

- 控制文件，如 CURRENT、IDENTITY 和 MANIFEST-000011。
- DB 表文件，如 004112.sst。
- 提前写入日志，如 000038.log。

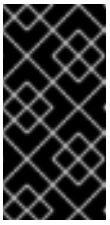
此外，还有一个内部的隐藏文件，充当 BlueFS replay 日志 (ino 1)， 它充当目录结构、文件映射和操作日志。

回退层次结构

使用 BlueFS，可以将任何文件放在任何设备上。文件的部分甚至可以位于不同的设备上，即 WAL、DB 和 SLOW。顺序是 BlueFS 放置文件。只有在主存储耗尽时，文件才会放入次要存储，仅当次要存储耗尽时。

特定文件的顺序是：

- 提前写入日志：WAL、DB、SLOW
- Replay log ino 1: DB, SLOW
- 控制和 DB 文件：DB、SLOW
  - 在用尽空间时控制和 DB 文件顺序：SLOW



重要

控制和 DB 文件顺序的例外。当 RocksDB 检测到您在 DB 文件上运行的空间不足时，它会直接通知您将文件放入 SLOW 设备。

11.8.1. 查看 bluefs_buffered_io 设置

作为存储管理员，您可以查看 bluefs_buffered_io 参数的当前设置。

对于 Red Hat Ceph Storage，默认将 bluefs_buffered_io 选项设为 True。这个选项使 BlueFS 能够在某些情况下执行缓冲的读取，并允许内核页面缓存作为辅助缓存进行读取，如 RocksDB 块读取。

重要

不建议更改 bluefs_buffered_io 的值。在更改 bluefs_buffered_io 参数前，请联系您的红帽支持团队。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。
- Ceph 监控节点的根级别访问权限.

流程

1. 登录到 Cephadm shell：

   示例

   ```
   [root@host01 ~]# cephadm shell
   ```

2. 您可以通过三种不同的方式查看 bluefs_buffered_io 参数的当前值：

方法 1

- 查看存储在配置数据库中的值：

示例

```
[ceph: root@host01 /]# ceph config get osd bluefs_buffered_io
```

方法 2

- 查看为特定 OSD 存储在配置数据库中的值：

语法

```
ceph config get OSD_ID bluefs_buffered_io
```

示例

```
[ceph: root@host01 /]# ceph config get osd.2 bluefs_buffered_io
```

方法 3

- 查看运行值的 OSD 的运行值与配置数据库中存储的值不同：

语法

```
ceph config show OSD_ID bluefs_buffered_io
```

示例

```
[ceph: root@host01 /]# ceph config show osd.3 bluefs_buffered_io
```

11.8.2. 查看 Ceph OSD 的 Ceph BlueFS 统计信息

使用 bluefs stats 命令查看并置和非并置 Ceph OSD 的与 BluesFS 相关的信息。

先决条件

- **一个正在运行的 Red Hat Ceph Storage 集群。**
- **对象存储配置为 BlueStore。**
- **对 OSD 节点的 root 级别访问权限。**

流程

1. **登录到 Cephadm shell：**

   示例

   ```
   [root@host01 ~]# cephadm shell
   ```

2. 查看 **BlueStore OSD** 统计信息：

语法

```
ceph daemon osd.OSD_ID bluefs stats
```

collocated OSD 示例

```
[ceph: root@host01 /]# ceph daemon osd.1 bluefs stats

1 : device size 0x3bfc00000 : using 0x1a428000(420 MiB)
wal_total:0, db_total:15296836403, slow_total:0
```

非并置 OSD 示例

```
[ceph: root@host01 /]# ceph daemon osd.1 bluefs stats

0 :
1 : device size 0x1dfbfe000 : using 0x1100000(17 MiB)
2 : device size 0x27fc00000 : using 0x248000(2.3 MiB)
RocksDBBlueFSVolumeSelector: wal_total:0, db_total:7646425907,
slow_total:10196562739, db_avail:935539507
Usage matrix:
DEV/LEV     WAL         DB          SLOW        *           *           REAL        FILES
LOG         0 B         4 MiB       0 B         0 B         0 B         756 KiB     1
WAL         0 B         4 MiB       0 B         0 B         0 B         3.3 MiB     1
DB          0 B         9 MiB       0 B         0 B         0 B         76 KiB      10
SLOW        0 B         0 B         0 B         0 B         0 B         0 B         0
TOTALS      0 B         17 MiB      0 B         0 B         0 B         0 B         12
MAXIMUMS:
LOG         0 B         4 MiB       0 B         0 B         0 B         756 KiB
WAL         0 B         4 MiB       0 B         0 B         0 B         3.3 MiB
DB          0 B         11 MiB      0 B         0 B         0 B         112 KiB
SLOW        0 B         0 B         0 B         0 B         0 B         0 B
TOTALS      0 B         17 MiB      0 B         0 B         0 B         0 B
```

其中：

0 ：这指的是专用的 WAL 设备，即 block.wal。

1： 这指的是专用的 DB 设备，即 block.db。

2： 这指的是主块设备，即 block 或 slow。

设备大小 ：它代表设备的实际大小。

使用: 表示总使用量。它不仅限于 BlueFS。

注意

DB 和 WAL 设备仅由 BlueFS 使用。对于主设备，也包含来自存储的 BlueStore 数据的使用量。在上例中，2.3 MiB 是 BlueStore 的数据。

wal_total,db_total,slow_total: 这些值重新迭代上面的设备值。

db_avail ：如果需要，这个值代表可以从 SLOW 设备获取多少字节。

使用列表

- rows WAL,DB,SLOW: 描述特定文件要放置的位置。
- 行 LOG ：描述 BlueFS replay log ino 1。
- WAL,DB,SLOW: 描述数据实际放置的位置。这些值以分配单位为单位。因为性能的原因， WAL 和 DB 有较大的分配单元。

- 列 * / *：与虚拟设备 new-db 和 new-wal 相关的虚拟设备，它们用于 ceph-bluestore-tool。它应始终显示 0 B。

- 列 REAL ：显示实际使用量（以字节为单位）。

- 列 FILES ：显示文件计数。

MAXIMUMS ：此表从用量列表中捕获每个条目的最大值。

其它资源

- 有关 BlueFS 文件的更多信息，请参阅 *Ceph BlueStore BlueFS*。

- 有关 *BlueStore 设备的更多信息，请参阅 Ceph* BlueStore 设备。

11.9. 使用 CEPH-BLUSTORE-TOOL

ceph-bluestore-tool 是一个在 BlueStore 实例上执行低级管理操作的实用程序。

以下命令可用于 ceph-bluestore-tool

语法

```
ceph-bluestore-tool COMMAND [ --dev DEVICE ... ] [ -i OSD_ID ] [ --path OSD_PATH ] [ --out-dir DIR ] [ --log-file | -l filename ] [ --deep ]

ceph-bluestore-tool fsck|repair --path OSD_PATH [ --deep ]

ceph-bluestore-tool qfsck --path OSD_PATH

ceph-bluestore-tool allocmap --path OSD_PATH

ceph-bluestore-tool restore_cfb --path OSD_PATH

ceph-bluestore-tool show-label --dev DEVICE ...
```

```
ceph-bluestore-tool prime-osd-dir --dev DEVICE --path OSD_PATH

ceph-bluestore-tool bluefs-export --path OSD_PATH --out-dir DIR

ceph-bluestore-tool bluefs-bdev-new-wal --path OSD_PATH --dev-target NEW_DEVICE

ceph-bluestore-tool bluefs-bdev-new-db --path OSD_PATH --dev-target NEW_DEVICE

ceph-bluestore-tool bluefs-bdev-migrate --path OSD_PATH --dev-target NEW_DEVICE --devs-source DEVICE1 [--devs-source DEVICE2]

ceph-bluestore-tool free-dump|free-score --path OSD_PATH [ --allocator block/bluefs-wal/bluefs-db/bluefs-slow ]

ceph-bluestore-tool reshard --path OSD_PATH --sharding NEW_SHARDING [ --sharding-ctrl CONTROL_STRING ]

ceph-bluestore-tool show-sharding --path OSD_PATH
```

每个 BlueStore 块设备在设备的开头都有一个块标签。您可以使用以下内容转储标签内容：

ceph-bluestore-tool show-label --dev *DEVICE*

主设备包含许多元数据，包括用于存储 OSD 数据目录中的小文件的信息。辅助设备(db 和 wal)只有最少的必填字段：OSD UUID、大小、设备类型和 birth 时间。

为 OSD 数据目录生成内容，以使用 prime-osd-dir 命令启动 BlueStore OSD。

ceph-bluestore-tool prime-osd-dir --dev *MAIN_DEVICE* -path /var/lib/ceph/osd/ceph-*ID*

表 11.2. ceph-bluestore-tool 命令

命令	描述
帮助	显示帮助
fsck [--deep]	选项： *on*,*off*; *yes*,*no*; *1*,*0*; 或 *true*,*false*.在 BlueStore 元数据上运行一致性检查。如果指定了 --deep，还会读取所有对象数据并验证校验和。
repair	运行一致性检查并修复任何错误。

命令	描述
qfsck	在 BlueStore 元数据与 ONodes 状态比较时运行一致性检查。分配器数据来自 RocksDB CFB （如果存在）以及 （如果没有使用 allocation-file ）。
Allocmap	执行 **qfsck** 完成的同一检查，然后存储一个新的 allocation-file。此命令默认为禁用，需要特殊构建。
restore_cfb	反向更改由新的 NCB 代码（通过 **ceph restart** 或运行 **allocmap** 命令时）完成，并恢复 RocksDB B Column-Family (allocator-map)。
bluefs-export	将 BlueFS 的内容导出到输出目录。
BlueFS-bdev-sizes --path *OSD_PATH*	将 BlueFS 理解的设备大小输出到 stdout。
bluefs-bdev-expand --path *OSD_PATH*	指示 BlueFS 检查其块设备的大小，如果扩展，则使用额外的空间。请注意，如果有足够可用空间，只有 BlueFS 创建的新文件才会在首选块设备上分配，而在 RocksDB 执行压缩时，对较慢的设备的现有文件将逐渐移除。换句话说，如果对较慢的设备存在任何数据中断，它将随着时间的推移移到快速设备中。
BlueFS-bdev-new-wal --path *OSD_PATH* --dev-target *NEW_DEVICE*	将 WAL 设备添加到 BlueFS，如果 WAL 设备已存在，则会失败。
BlueFS-bdev-new-db --path *OSD_PATH* --dev-target *NEW_DEVICE*	将 DB 设备添加到 BlueFS，如果 DB 设备已存在，则会失败。
BlueFS-bdev-migrate --dev-target *NEW_DEVICE* --devs-source *DEVICE1* [--devs-source *DEVICE2*]	将 BlueFS 数据从源设备移动到目标，源设备除外（主设备除外）成功删除。目标设备可以同时附加或者新设备。在后者的情况下，它被添加到 OSD 中，替换其中一个源设备。适用以下替换规则（按优先级顺序，在第一个匹配项中停止）：(1)如果源列表有 DB volume - 目标设备替换它。(2)如果源列表有 WAL 卷 - 目标设备替换它。(3)如果源列表只具有较慢的卷 - 不允许操作，需要通过 new-db/new-wal 命令显式分配。
show-label --dev *DEVICE* [...]	显示任何设备标签。
free-dump --path *OSD_PATH* [--allocator block/bluefs-wal/bluefs-db/bluefs-slow]	转储分配器中的所有空闲区域。
free-score --path *OSD_PATH* [--allocator block/bluefs-wal/bluefs-db/bluefs-slow]	提供一个 [0-1] 号，代表分配器中的碎片。0 代表当所有空闲空间都位于一个块中时。1 代表最糟糕的碎片。

命令	描述
reshard --path *OSD_PATH* -sharding *NEW_SHARDING* [--resharding-ctrl *CONTROL_STRING*]	更改 BlueStore 的 RocksDB 分片。分片基于 RocksDB 列系列构建。此选项允许测试新分片的性能，而无需重新部署 OSD。重新划分过程通常是较长的过程，它涉及遍历整个 RocksDB 密钥空间，并将其部分移到不同的列中。**--resharding-ctrl** 选项提供对重新划分过程的性能控制。中断重新划分将阻止 OSD 运行。中断的重新划分不会损坏数据。始终可以继续重新划分，或者选择任何其他分片方案，包括恢复到原始分片方案。有关重新划分的更多信息，请参阅使用 BlueStore 管理工具重新划分 RocksDB 数据库的 *OSD* 部分。
show-sharding --path *OSD_PATH*	显示当前应用到 BlueStore RocksDB 的分片。

表 11.3. ceph-bluestore-tool 命令选项

命令选项	描述
--dev *DEVICE*	将设备添加到要考虑的设备列表中。
-I *OSD_ID*	作为 OSD *OSD_ID* 进行操作。连接到监控 OSD 特定选项。如果 monitor 不可用，请添加 **--no-mon-config** 来从 ceph.conf 读取。
--Devs-source *DEVICE*	将设备添加到设备列表中，以被视为迁移源。
--dev-target *DEVICE*	*指定用于* 添加新 DB/WAL 的目标设备迁移操作或设备。
--path *OSD_PATH*	指定 OSD 路径。在大多数情况下，设备列表是从 osd path 中存在的符号链接中推断出来的。这通常比使用 --dev 明确指定设备更简单。如果提供了 if **-i osd_id**，则不需要这个选项。
--out-dir *DIR*	**bluefs-export** 的输出目录。
-l,--log-file *LOG_FILE*	要记录到的文件。
--log-level *NUM*	调试日志级别。默认值为 30 （冗余），20 个非常详细，10 非常详细，1 并不是非常详细。
--deep	深度清理/修复（读取和验证对象数据，而不仅限于元数据）。
--allocator *NAME*	对于 free-dump 和 free-score 操作非常有用。选择分配器。

命令选项	描述
--resharding-ctrl *CONTROL_STRING*	提供对重新划分过程的控制。指定 RocksDB 迭代器的频率，以及在提交到 RocksDB 前应如何提交批处理。选项格式为： **<iterator_refresh_bytes>/<iterator_refresh_keys>/<batch_commit_bytes>/<batch_commit_keys>** 默认值：10000000/10000/1000000/1000

流程

1. **在使用 ceph-bluestore-tool 之前停止 OSD。**

   语法

   ```
   ceph orch daemon stop osd.ID
   ```

   示例

   ```
   [ceph: root@host01 /]# ceph orch daemon stop osd.2
   ```

2. **从 OSD 节点，登录目标 OSD 容器。**

   语法

   ```
   cephadm shell --name osd.ID
   ```

示例

```
[ceph: root@host01 /]# ceph shell --name osd.2
```

3. 运行所需命令。

示例

```
[ceph: root@host01 /]# ceph-bluestore-tool bluefs-bdev-new-wal --dev-target /dev/test/newdb
--path /var/lib/ceph/osd/ceph-0
```

注意

本例演示了添加新的 wal 设备。

4. **从 cephadm shell，重启 OSD。**

语法

```
ceph orch daemon start osd.ID
```

示例

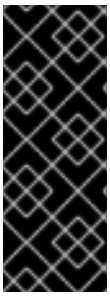
```
[ceph: root@host01 /]# ceph orch daemon start osd.2
```

其它资源

有关 *BlueStore 配置选项* 的详情，请参阅 BlueStore 配置选项。

第 12 章 CRIMSON （技术预览）

作为存储管理员，Crimson 项目是构建替代 ceph-osd 守护进程的努力，它适用于低延迟、高吞吐量持久内存和 NVMe 技术的新现实。



重要

Crimson 功能只是一个技术预览功能。红帽产品服务级别协议（SLA）不支持技术预览功能，且其功能可能并不完善，因此红帽不建议在生产环境中使用它们。这些技术预览功能可以使用户提早试用新的功能，并有机会在开发阶段提供反馈意见。如需了解更多详细信息，请参阅红帽技术预览功能的支持范围。

12.1. CRIMSON 概述

Crimson 是 crimson-osd 的代码名称，它是下一代用于多核心可扩展性的 ceph-osd。它通过快速网络和存储设备提高性能，采用包括 DPDK 和 SPDK 的顶级技术。BlueStore 继续支持 HDD 和 SSD。Crimson 旨在与早期版本的 OSD 守护进程与类 ceph-osd 兼容。

Crimson 基于 SeaStar C++ 框架构建，是核心 Ceph 对象存储守护进程(OSD)组件的新实现，并替换了 ceph-osd。crimson-osd 最小化延迟并增加 CPU 处理器用量。它使用高性能异步 IO 和新的线程架构，旨在最小化上下文切换和用于跨通信的操作间的线程通信。

小心

对于 Red Hat Ceph Storage 7，您可以使用 Crimson 在复制池中测试 RADOS 块设备(RBD)工作负载。不要将 Crimson 用于生产环境数据。

Crimson 目标

Crimson OSD 是 OSD 守护进程的替代品，其目标如下：

最小化 CPU 超载

- 最小化周期或 IOPS。
- 最小化跨核心通信。

- 最小化副本。
- 绕过内核，避免上下文切换。

启用新兴存储技术

- 区命名空间
- 持久性内存
- 快速 NVMe

Seastar 功能

- 每个 CPU 的单一响应器线程
- 异步 IO
- 在用户空间中进行调度
- 包括对 DPDK 的直接支持，这是用户空间网络的高性能库。

优点

- SeaStore 具有独立的元数据集合。
- 事务
- 由扁平对象命名空间组成。

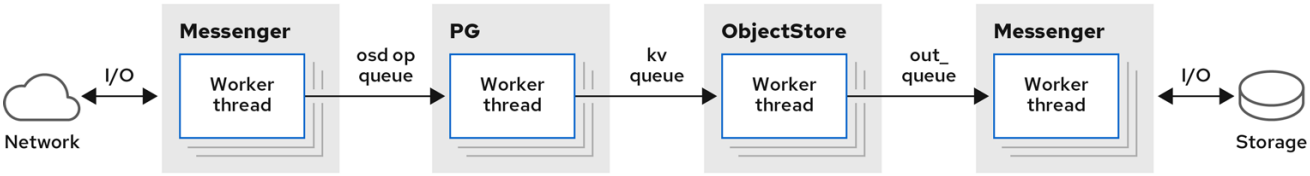
- 对象名称可能是大容量(>1k)。
- 每个对象包含一个 key>value 映射（字符串>字节）和数据有效负载。
- 支持 COW 对象克隆。
- 支持对 OMAP 和对象命名空间的排序列表。

12.2. CRIMSON 和 CLASSIC CEPH OSD 架构之间的区别

在典型的 ceph-osd 架构中，messenger 线程从 wire 读取客户端消息，它将消息放在 OP 队列中。然后，osd-op thread-pool 会提取消息，并创建一个事务并将其排队到 BlueStore，当前的默认 ObjectStore 实现。然后，BlueStore 的 kv_queue 会获取这个事务，以及队列中的任何其他事务，同步等待 rocksdb 提交事务，然后将完成回调放在完成队列中。然后，完成程序线程会获取完成回调和队列，以取代要发送的 messenger 线程。

每个操作都需要对队列的内容进行间线程协调。对于 pg state，可能有多个线程会需要访问 PG 的内部元数据来锁定争用。

随着处理器使用量增加的增加，处理器使用率会快速扩展任务和内核数量，每个锁定点可能会成为某些情况下的扩展瓶颈。此外，即使未延续，这些锁定和队列也会产生延迟成本。由于此延迟，线程池和任务队列 deteriorate，因为工作线程和锁定之间的委派任务可以强制上下文切换。



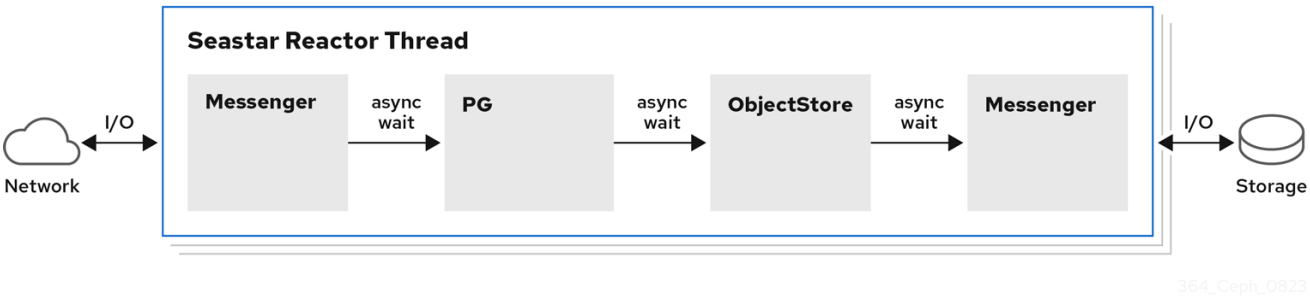
与 ceph-osd 架构不同，Crimson 允许单个 I/O 操作在没有上下文切换的情况下在单一内核中完成，而不阻止底层存储操作不需要它。但是，有些操作仍需要等待异步进程完成，可能根据恢复或底层设备等系统状态，不确定。

Crimson 使用称为 Seastar 的 C++ 框架，这是一个高度异步引擎，通常预先分配一个线程固定到每个内核。它们在这些内核间划分工作，以便避免在内核和锁定之间对状态进行分区。使用 Seastar 时，I/O 操作会根据目标对象在一组线程中进行分区。除了在不同线程组之间分割运行 I/O 操作的不同阶段，请在

单个线程中运行所有管道阶段。如果需要阻止某个操作，内核的 Seastar 反应器会切换到另一个并发操作和进度。

理想情况下，不再需要所有锁定和上下文切换，因为每个运行非阻塞任务拥有 CPU，直到它完成或合作生成为止。其他线程无法同时抢占任务。如果数据路径中的其他分片不需要通信，则理想的性能会线性扩展内核数量，直到 I/O 设备达到其限制为止。此设计适合 Ceph OSD，因为在 OSD 级别上，PG 分片所有 IO。

与 ceph-osd 不同，crimson-osd 不会自行守护进程，即使启用了 daemonize 选项。不要守护进程化 crimson-osd，因为支持的 Linux 发行版使用 systemd，它可以对应用程序进行守护进程化。使用 sysvinit 时，使用 start-stop-daemon 来守护进程化 crimson-osd。



ObjectStore 后端

crimson-osd 提供原生和 alienized 对象存储后端。原生对象存储后端使用 Seastar 响应器执行 I/O。

Crimson 支持以下三个 ObjectStore 后端：

- AlienStore - 提供与早期版本的对象存储（即 BlueStore）的兼容性。
- CyanStore - 用于测试的模拟后端，由易失性内存实施。此对象存储在典型的 OSD 中的 memstore 后建模。
- SeaStore - 为 Crimson OSD 设计的新对象存储。对多个分片支持的路径因后端的特定目标而异。

以下是其他两个典型的 OSD ObjectStore 后端：

- MemStore - 作为后端对象存储的内存。
- BlueStore - 类 ceph-osd 使用的对象存储。

12.3. CRIMSON 指标

Crimson 有三种方法来报告统计信息和指标：

- PG 统计报告给管理器。
- Prometheus 文本协议。
- asock 命令。

PG 统计报告到管理器

Crimson 收集 per-pg、per-pool，以及 MPGStats 消息中的 per-osd stats （发送到 Ceph 管理器）。

Prometheus 文本协议

使用 --prometheus-port 命令行选项配置侦听端口和地址。

asock 命令

提供的管理 socket 命令用于转储指标。

语法

```
ceph tell OSD_ID dump_metrics
ceph tell OSD_ID dump_metrics reactor_utilization
```

示例

```
[ceph: root@host01 /]# ceph tell osd.0 dump_metrics
[ceph: root@host01 /]# ceph tell osd.0 dump_metrics reactor_utilization
```

在这里，**reactor_utilization** 是一个可选字符串，用于按前缀过滤转储的指标。

12.4. CRIMSON 配置选项

为 **Seastar** 特定的命令行选项运行 **crimson-osd --help-seastar** 命令。以下是可用于配置 **Crimson** 的选项：

--crimson, 描述

启动 **crimson-osd** 而不是 **ceph-osd**。

--nodaemon, 描述

不要守护进程化服务。

--redirect-output, 描述

将 **stdout** 和 **stderr** 重定向到 **out/$type.$num.stdout**

--osd-args, 描述

将额外的命令行选项传递给 **crimson-osd** 或 **ceph-osd**。这个选项对于将 **Seastar** 选项传给 **crimson-osd** 非常有用。例如，可以提供 **--osd-args "--memory 2G"** 来设定要使用的内存量。

--cyanstore, 描述

使用 **CyanStore** 作为对象存储后端。

--bluestore, 描述

使用 alienized BlueStore 作为对象存储后端。-- BlueStore 是默认的内存存储。

--memstore, 描述

使用 alienized MemStore 作为对象存储后端。

--seastore, 描述

使用 SeaStore 作为后端对象存储。

--seastore-devs, 描述

指定 SeaStore 使用的块设备。

--seastore-secondary-devs, 描述

可选。SeaStore 支持多个设备。通过将块设备传递给这个选项来启用此功能。

--seastore-secondary-devs-type, 描述

可选。指定辅助设备的类型。当辅助设备比传递给 --seastore-devs 的主设备慢时，更快的设备中的冷数据将随着时间推移而被逐出到较慢的设备中。有效类型包括 HDD、SSD、（默认）、ZNS 和 RANDOM_BLOCK_SSD。请注意，从设备不应比主设备快。

12.5. 配置 CRIMSON

通过安装新存储集群来配置 crimson-osd。使用 bootstrap 选项安装新集群。您无法升级此集群，因为它处于实验性阶段。警告：请勿使用生产数据，因为它可能会导致数据丢失。

先决条件

- 第一个 Ceph 监控容器的 IP 地址，也是存储集群中第一个节点的 IP 地址。
- 登录到 registry.redhat.io。
- 至少 10 GB 的可用空间用于 /var/lib/containers/。
- 所有节点的根级别访问权限。

流程

1. **在 bootstrap 时，使用 --image 标志来使用 Crimson 构建。**

   示例

   ```
   [root@host 01 ~]# cephadm --image quay.ceph.io/ceph-
   ci/ceph:b682861f8690608d831f58603303388dd7915aa7-crimson bootstrap --mon-ip
   10.1.240.54 --allow-fqdn-hostname --initial-dashboard-password Ceph_Crims
   ```

2. **登录到 cephadm shell：**

   示例

   ```
   [root@host 01 ~]# cephadm shell
   ```

3. **全局启用 Crimson 作为实验性功能。**

   示例

   ```
   [ceph: root@host01 /]# ceph config set global
   'enable_experimental_unrecoverable_data_corrupting_features' crimson
   ```

   此步骤启用 crimson。Crimson 是高度实验性的，需要出现故障，包括崩溃和数据丢失。

4. 启用 **OSD map** 标志。

示例

```
[ceph: root@host01 /]# ceph osd set-allow-crimson --yes-i-really-mean-it
```

monitor 允许 **crimson-osd** 仅使用 **--yes-i-really-mean-it** 标志引导。

5. 为 **monitor** 启用 **Crimson** 参数，以指示将默认池创建为 **Crimson** 池。

示例

```
[ceph: root@host01 /]#  ceph config set mon osd_pool_default_crimson true
```

crimson-osd 没有为非**crimson** 池启动放置组**(PG)**。

12.6. CRIMSON 配置参数

以下是可用于配置 **Crimson** 的参数。

crimson_osd_obc_lru_size

描述

要缓存的 **obcs** 数量。

类型

uint

Default（默认）

10

crimson_osd_scheduler_concurrency

描述

并发 IO 操作的最大数量，0 代表无限。

类型

uint

Default（默认）

0

crimson_alien_op_num_threads

描述

提供 alienized ObjectStore 的线程数量。

类型

uint

Default（默认）

6

crimson_seastar_smp

描述

用于 OSD 的 seastar 响应器线程的数量。

类型

uint

Default（默认）

1

crimson_alien_thread_cpu_cores

描述

一个lienstore 线程以 cpuset (7)格式运行的字符串 CPU 内核。

类型

字符串

seastore_segment_size

描述

用于分段管理器的片段大小。

类型

大小

Default（默认）

64_M

seastore_device_size

描述

创建时用于 SegmentManager 块文件的总大小。

类型

大小

Default（默认）

50_G

seastore_block_create

描述

如果不存在，请创建 SegmentManager 文件。

类型

布尔值

Default（默认）

true

seastore_journal_batch_capacity

描述

日志批处理中的记录数量限制。

类型

uint

Default（默认）

16

seastore_journal_batch_flush_size

描述

强制清除日志批处理的大小阈值。

类型

大小

Default（默认）

16_M

seastore_journal_iodepth_limit

描述

用于提交日志记录的 IO 深度限制。

类型

uint

Default（默认）

5

seastore_journal_batch_preferred_fullness

描述

清除日志批处理的记录完整阈值。

类型

浮点值

Default（默认）

0.95

seastore_default_max_object_size

描述

seastore 对象数据的默认逻辑地址空间保留。

类型

uint

Default（默认）

16777216

seastore_default_object_metadata_reservation

描述

seastore 对象的元数据的默认逻辑地址空间保留。

类型

uint

Default（默认）

16777216

seastore_cache_lru_size

描述

要保留在缓存中的扩展大小（以字节为单位）。

类型

大小

Default（默认）

64_M

seastore_cbjournal_size

描述

创建时用于 CircularBoundedJournal 的总大小，只有在 seastore_main_device_type 是 RANDOM_BLOCK 时有效。

类型

大小

Default（默认）

5_G

seastore_obj_data_write_amplification

描述

如果写入大小的总扩展大小超过这个值，则分割扩展。

类型

浮点值

Default（默认）

1.25

seastore_max_concurrent_transactions

描述

seastore 允许的最大并发事务。

类型

uint

Default（默认）

8

seastore_main_device_type

描述

主设备类型 seastore 使用(SSD 或 RANDOM_BLOCK_SSD)。

类型

字符串

Default（默认）

SSD

seastore_multiple_tiers_stop_evict_ratio

描述

当主层使用的比率小于这个值时，停止将冷数据驱除到冷层。

类型

浮点值

Default（默认）

0.5

seastore_multiple_tiers_default_evict_ratio

描述

在使用主层使用比率达到这个值时，开始将冷数据驱除到冷层。

类型

浮点值

Default（默认）

0.6

seastore_multiple_tiers_fast_evict_ratio

描述

当主层使用比率达到这个值时，立即开始驱除。

类型

浮点值

Default（默认）

0.7

12.7. 分析 CRIMSON

分析 Crimson 是利用 Crimson 进行性能测试的方法。支持两种类型的性能分析：

- 灵活的 I/O (FIO)- crimson-store-nbd 将可配置 FuturizedStore 内部显示为用于 FIO 的 NBD 服务器。
- Ceph 基准测试工具(CBT)- 在 python 中测试利用，测试 Ceph 集群的性能。

流程

1. 安装 libnbd 和编译 FIO：

示例

```
[root@host01 ~]# dnf install libnbd
[root@host01 ~]# git clone git://git.kernel.dk/fio.git
[root@host01 ~]# cd fio
[root@host01 ~]# ./configure --enable-libnbd
[root@host01 ~]# make
```

2. build crimson-store-nbd:

示例

```
[root@host01 ~]# cd build
[root@host01 ~]# ninja crimson-store-nbd
```

3. **使用块设备运行 crimson-store-nbd 服务器。指定原始设备的路径，如 /dev/nvme1n1 ：**

示例

```
[root@host01 ~]# export disk_img=/tmp/disk.img
[root@host01 ~]# export unix_socket=/tmp/store_nbd_socket.sock
[root@host01 ~]# rm -f $disk_img $unix_socket
[root@host01 ~]# truncate -s 512M $disk_img
[root@host01 ~]# ./bin/crimson-store-nbd \
  --device-path $disk_img \
  --smp 1 \
  --mkfs true \
  --type transaction_manager \
  --uds-path ${unix_socket} &
 --smp is the CPU cores.
--mkfs initializes the device first.
--type is the backend.
```

4. **创建名为 nbd.fio 的 FIO 作业：**

示例

```
[global]
ioengine=nbd
uri=nbd+unix:///?socket=${unix_socket}
rw=randrw
time_based
runtime=120
group_reporting
iodepth=1
size=512M
```

```
[job0]
offset=0
```

5. **使用编译的 FIO 测试 Crimson 对象：**

示例

```
[root@host01 ~]# ./fio nbd.fio
```

Ceph 基准工具(CBT)

针对两个分支运行相同的测试。一个是 main(master)，另一个是您选择的 主题 分支。比较测试结果。除了每个测试案例外，还会定义一组规则来检查在比较两组测试结果时是否需要执行回归。如果找到可能的回归问题，则会突出显示规则和对应的测试结果。

流程

1. **在主分支和主题分支中运行 make crimson osd ：**

示例

```
[root@host01 ~]# git checkout master
[root@host01 ~]# make crimson-osd
[root@host01 ~]# ../src/script/run-cbt.sh --cbt ~/dev/cbt -a /tmp/baseline
../src/test/crimson/cbt/radosbench_4K_read.yaml
[root@host01 ~]# git checkout topic
[root@host01 ~]# make crimson-osd
[root@host01 ~]# ../src/script/run-cbt.sh --cbt ~/dev/cbt -a /tmp/yap
../src/test/crimson/cbt/radosbench_4K_read.yaml
```

2. **比较测试结果：**

示例

```
[root@host01 ~]# ~/dev/cbt/compare.py -b /tmp/baseline -a /tmp/yap -v
```

第 13 章 CEPHADM 故障排除

作为存储管理员，您可以对 Red Hat Ceph Storage 集群进行故障排除。有时需要调查 Cephadm 命令失败的原因，或者为何特定的服务未正确运行。

13.1. 暂停或禁用 CEPHADM

如果 Cephadm 没有按预期工作，您可以使用以下命令暂停大多数后台活动：

示例

```
[ceph: root@host01 /]# ceph orch pause
```

这将停止任何更改，但 Cephadm 定期检查主机以刷新守护进程和设备的清单。

如果要完全禁用 Cephadm，请运行以下命令：

示例

```
[ceph: root@host01 /]# ceph orch set backend ''
[ceph: root@host01 /]# ceph mgr module disable cephadm
```

请注意，之前部署的守护进程容器仍然存在，并在之前启动它们。

要在集群中重新启用 Cephadm，请运行以下命令：

示例

```
[ceph: root@host01 /]# ceph mgr module enable cephadm
[ceph: root@host01 /]# ceph orch set backend cephadm
```

13.2. 每个服务和每个守护进程事件

Cephadm 在每个服务以及每个守护进程之间存储事件，以帮助调试失败的守护进程部署。这些事件通常包含相关信息：

针对**每个服务**

语法

```
ceph orch ls --service_name SERVICE_NAME --format yaml
```

示例

```
[ceph: root@host01 /]# ceph orch ls --service_name alertmanager --format yaml
service_type: alertmanager
service_name: alertmanager
placement:
  hosts:
  - unknown_host
status:
  ...
  running: 1
  size: 1
events:
- 2021-02-01T08:58:02.741162 service:alertmanager [INFO] "service was created"
- '2021-02-01T12:09:25.264584 service:alertmanager [ERROR] "Failed to apply: Cannot
  place <AlertManagerSpec for service_name=alertmanager> on unknown_host: Unknown hosts"'
```

针对每个守护进程

语法

```
ceph orch ps --service-name SERVICE_NAME --daemon-id DAEMON_ID --format yaml
```

示例

```
[ceph: root@host01 /]# ceph orch ps --service-name mds --daemon-id cephfs.hostname.ppdhsz --
format yaml
daemon_type: mds
daemon_id: cephfs.hostname.ppdhsz
hostname: hostname
status_desc: running
...
events:
- 2021-02-01T08:59:43.845866 daemon:mds.cephfs.hostname.ppdhsz [INFO] "Reconfigured
  mds.cephfs.hostname.ppdhsz on host 'hostname'"
```

13.3. 检查 CEPHADM 日志

您可以使用以下命令实时监控 Cephadm 日志：

示例

```
[ceph: root@host01 /]# ceph -W cephadm
```

您可以使用以下命令查看最后几个信息：

示例

```
[ceph: root@host01 /]# ceph log last cephadm
```

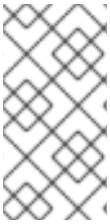
如果您启用了对文件的日志，则可以查看 monitor 主机上名为 ceph.cephadm.log 的 Cephadm 日志文件。

13.4. 收集日志文件

您可以使用 journalctl 命令为所有守护进程收集日志文件。

注意

您必须在 cephadm shell 之外运行所有这些命令。



注意

默认情况下，Cephadm 将日志存储在 journald 中，这意味着守护进程日志在 /var/log/ceph 中不再可用。

- **要读取特定守护进程的日志文件，请运行以下命令：**

语法

```
cephadm logs --name DAEMON_NAME
```

示例

```
[root@host01 ~]# cephadm logs --name cephfs.hostname.ppdhsz
```

注意

在运行守护进程的同一主机上运行时，此命令可以正常工作。

- 要读取在不同主机上运行的特定守护进程的日志文件，请运行以下命令：

语法

```
cephadm logs --fsid FSID --name DAEMON_NAME
```

示例

```
[root@host01 ~]# cephadm logs --fsid 2d2fd136-6df1-11ea-ae74-002590e526e8 --name
cephfs.hostname.ppdhsz
```

其中 **fsid** 是 **ceph status** 命令提供的集群 ID。

- 要获取给定主机上所有守护进程的所有日志文件，请运行以下命令：

语法

```
for name in $(cephadm ls | python3 -c "import sys, json; [print(i['name']) for i in
json.load(sys.stdin)]") ; do cephadm logs --fsid FSID_OF_CLUSTER --name "$name" >
$name; done
```

示例

```
[root@host01 ~]# for name in $(cephadm ls | python3 -c "import sys, json; [print(i['name']) for i in json.load(sys.stdin)]") ; do cephadm logs --fsid 57bddb48-ee04-11eb-9962-001a4a000672 --name "$name" > $name; done
```

13.5. 收集 SYSTEMD 状态

- **要输出 systemd 单元的状态，请运行以下命令：**

示例

```
[root@host01 ~]$ systemctl status ceph-a538d494-fb2a-48e4-82c8-b91c37bb0684@mon.host01.service
```

13.6. 列出所有下载的容器镜像

要列出主机上下载的所有容器镜像，请运行以下命令：

示例

```
[ceph: root@host01 /]# podman ps -a --format json | jq '.[].Image'
"docker.io/library/rhel9"
"registry.redhat.io/rhceph-alpha/rhceph-6-rhel9@sha256:9aaea414e2c263216f3cdcb7a096f57c3adf6125ec9f4b0f5f65fa8c43987155"
```

13.7. 手动运行容器

Cephadm 编写运行容器的小型打包程序。请参阅 /var/lib/ceph/*CLUSTER_FSID*/*SERVICE_NAME*/unit 来运行容器执行命令。

SSH 错误

如果遇到以下错误：

示例

```
execnet.gateway_bootstrap.HostNotFound: -F /tmp/cephadm-conf-73z09u6g -i /tmp/cephadm-identity-ky7ahp_5 root@10.10.1.2
...
raise OrchestratorError(msg) from e
orchestrator._interface.OrchestratorError: Failed to connect to 10.10.1.2 (10.10.1.2).
Please make sure that the host is reachable and accepts connections using the cephadm SSH key
```

尝试排除这个问题的选项：

- 为确保 Cephadm 具有 SSH 身份密钥，请运行以下命令：

  示例

  ```
  [ceph: root@host01 /]# ceph config-key get mgr/cephadm/ssh_identity_key > ~/cephadm_private_key
  INFO:cephadm:Inferring fsid f8edc08a-7f17-11ea-8707-000c2915dd98
  INFO:cephadm:Using recent ceph image docker.io/ceph/ceph:v15 obtained 'mgr/cephadm/ssh_identity_key'
  [root@mon1 ~] # chmod 0600 ~/cephadm_private_key
  ```

  如果上述命令失败，Cephadm 没有密钥。要生成 SSH 密钥，请运行以下命令：

示例

```
[ceph: root@host01 /]# chmod 0600 ~/cephadm_private_key
```

或者

示例

```
[ceph: root@host01 /]# cat ~/cephadm_private_key | ceph cephadm set-ssk-key -i-
```

- **要确保 SSH 配置正确，请运行以下命令：**

示例

```
[ceph: root@host01 /]# ceph cephadm get-ssh-config
```

- **要验证与主机的连接，请运行以下命令：**

示例

```
[ceph: root@host01 /]# ssh -F config -i ~/cephadm_private_key root@host01
```

验证公钥是否在 authorized_keys 中。

要验证公钥是否在 authorized_keys 文件中，请运行以下命令：

示例

```
[ceph: root@host01 /]# ceph cephadm get-pub-key
[ceph: root@host01 /]# grep "`cat ~/ceph.pub`" /root/.ssh/authorized_keys
```

13.8. CIDR 网络错误

无类别域路由(CIDR)也称为超级网络域路由，是一种分配互联网协议(IP)地址的方法，Fthe Cephadm 日志条目显示当前状态，以基于类 A、类 B 和 Class C 网络来提高了地址分发效率，并替换之前系统。如果您看到以下错误之一：

```
ERROR: Failed to infer CIDR network for mon ip *; pass --skip-mon-network to configure it later
```

或者

```
Must set public_network config option or specify a CIDR network, ceph addrvec, or plain IP
```

您需要运行以下命令：

示例

```
[ceph: root@host01 /]# ceph config set host public_network hostnetwork
```

13.9. 访问管理 SOCKET

每个 Ceph 守护进程都提供一个可绕过 MON 的管理 socket。

要访问管理套接字，请输入主机上的守护进程容器：

示例

```
[ceph: root@host01 /]# cephadm enter --name cephfs.hostname.ppdhsz
[ceph: root@mon1 /]# ceph --admin-daemon /var/run/ceph/ceph-cephfs.hostname.ppdhsz.asok
config show
```

13.10. 手动部署 MGR 守护进程

Cephadm 需要 mgr 守护进程来管理 Red Hat Ceph Storage 集群。如果删除了 Red Hat Ceph Storage 集群的最后一个 mgr 守护进程，您可以在 Red Hat Ceph Storage 集群随机主机上手动部署 mgr 守护进程。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。
- 所有节点的根级别访问权限。
- 主机添加到集群中。

流程

1. 登录到 Cephadm shell：

   示例

   ```
   [root@host01 ~]# cephadm shell
   ```

2. **禁用 Cephadm 调度程序，以使用以下命令防止 Cephadm 删除新的 MGR 守护进程：**

示例

```
[ceph: root@host01 /]# ceph config-key set mgr/cephadm/pause true
```

3. **获取或为新 MGR 守护进程创建 auth 条目：**

示例

```
[ceph: root@host01 /]# ceph auth get-or-create mgr.host01.smfvfd1 mon "profile mgr" osd
"allow *" mds "allow *"
[mgr.host01.smfvfd1]
key = AQDhcORgW8toCRAAlMzlqWXnh3cGRjqYEa9ikw==
```

4. **打开 ceph.conf 文件：**

示例

```
[ceph: root@host01 /]# ceph config generate-minimal-conf
# minimal ceph.conf for 8c9b0072-67ca-11eb-af06-001a4a0002a0
[global]
fsid = 8c9b0072-67ca-11eb-af06-001a4a0002a0
mon_host = [v2:10.10.200.10:3300/0,v1:10.10.200.10:6789/0]
[v2:10.10.10.100:3300/0,v1:10.10.200.100:6789/0]
```

5. **获取容器镜像：**

示例

```
[ceph: root@host01 /]# ceph config get "mgr.host01.smfvfd1" container_image
```

6. **创建 config-json.json 文件并添加以下内容：**

注意

使用 ceph config generate-minimal-conf 命令的输出中的值。

示例

```
{
 {
  "config": "# minimal ceph.conf for 8c9b0072-67ca-11eb-af06-001a4a0002a0\n[global]\n\tfsid
= 8c9b0072-67ca-11eb-af06-001a4a0002a0\n\tmon_host =
[v2:10.10.200.10:3300/0,v1:10.10.200.10:6789/0]
[v2:10.10.10.100:3300/0,v1:10.10.200.100:6789/0]\n",
  "keyring": "[mgr.Ceph5-2.smfvfd1]\n\tkey =
AQDhcORgW8toCRAAlMzlqWXnh3cGRjqYEa9ikw==\n"
}
}
```

7. **从 Cephadm shell 退出：**

示例

```
[ceph: root@host01 /]# exit
```

8. **部署 MGR 守护进程：**

示例

```
[root@host01 ~]# cephadm --image registry.redhat.io/rhceph-alpha/rhceph-6-rhel9:latest deploy --fsid  8c9b0072-67ca-11eb-af06-001a4a0002a0 --name mgr.host01.smfvfd1 --config-json config-json.json
```

验证

在 Cephadm shell 中，运行以下命令：

示例

```
[ceph: root@host01 /]# ceph -s
```

您可以看到新的 mgr 守护进程已被添加。

第 14 章 CEPHADM 操作

作为存储管理员，您可以在 Red Hat Ceph Storage 集群中执行 Cephadm 操作。

14.1. 监控 CEPHADM 日志消息

Cephadm 日志到 cephadm 集群日志频道，以便您可以实时监控进度。

- 要实时监控进度，请运行以下命令：

示例

```
[ceph: root@host01 /]# ceph -W cephadm
```

示例

```
2022-06-10T17:51:36.335728+0000 mgr.Ceph5-1.nqikfh [INF] refreshing Ceph5-adm facts
2022-06-10T17:51:37.170982+0000 mgr.Ceph5-1.nqikfh [INF] deploying 1 monitor(s) instead
of 2 so monitors may achieve consensus
2022-06-10T17:51:37.173487+0000 mgr.Ceph5-1.nqikfh [ERR] It is NOT safe to stop
['mon.Ceph5-adm']: not enough monitors would be available (Ceph5-2) after stopping mons
[Ceph5-adm]
2022-06-10T17:51:37.174415+0000 mgr.Ceph5-1.nqikfh [INF] Checking pool "nfs-ganesha"
exists for service nfs.foo
2022-06-10T17:51:37.176389+0000 mgr.Ceph5-1.nqikfh [ERR] Failed to apply nfs.foo spec
NFSServiceSpec({'placement': PlacementSpec(count=1), 'service_type': 'nfs', 'service_id':
'foo', 'unmanaged': False, 'preview_only': False, 'pool': 'nfs-ganesha', 'namespace': 'nfs-ns'}):
Cannot find pool "nfs-ganesha" for service nfs.foo
Traceback (most recent call last):
  File "/usr/share/ceph/mgr/cephadm/serve.py", line 408, in _apply_all_services
    if self._apply_service(spec):
  File "/usr/share/ceph/mgr/cephadm/serve.py", line 509, in _apply_service
    config_func(spec)
  File "/usr/share/ceph/mgr/cephadm/services/nfs.py", line 23, in config
    self.mgr._check_pool_exists(spec.pool, spec.service_name())
  File "/usr/share/ceph/mgr/cephadm/module.py", line 1840, in _check_pool_exists
    raise OrchestratorError(f'Cannot find pool "{pool}" for '
orchestrator._interface.OrchestratorError: Cannot find pool "nfs-ganesha" for service nfs.foo
2022-06-10T17:51:37.179658+0000 mgr.Ceph5-1.nqikfh [INF] Found osd claims -> {}
```

```
2022-06-10T17:51:37.180116+0000 mgr.Ceph5-1.nqikfh [INF] Found osd claims for
drivegroup all-available-devices -> {}
2022-06-10T17:51:37.182138+0000 mgr.Ceph5-1.nqikfh [INF] Applying all-available-devices
on host Ceph5-adm...
2022-06-10T17:51:37.182987+0000 mgr.Ceph5-1.nqikfh [INF] Applying all-available-devices
on host Ceph5-1...
2022-06-10T17:51:37.183395+0000 mgr.Ceph5-1.nqikfh [INF] Applying all-available-devices
on host Ceph5-2...
2022-06-10T17:51:43.373570+0000 mgr.Ceph5-1.nqikfh [INF] Reconfiguring node-
exporter.Ceph5-1 (unknown last config time)...
2022-06-10T17:51:43.373840+0000 mgr.Ceph5-1.nqikfh [INF] Reconfiguring daemon node-
exporter.Ceph5-1 on Ceph5-1
```

- **默认情况下，日志会显示 info 级别的事件及以上。要查看调试级别信息，请运行以下命令：**

示例

```
[ceph: root@host01 /]# ceph config set mgr mgr/cephadm/log_to_cluster_level debug
[ceph: root@host01 /]# ceph -W cephadm --watch-debug
[ceph: root@host01 /]# ceph -W cephadm --verbose
```

- **将调试级别恢复为默认 info ：**

示例

```
[ceph: root@host01 /]# ceph config set mgr mgr/cephadm/log_to_cluster_level info
```

- **要查看最近的事件，请运行以下命令：**

示例

```
[ceph: root@host01 /]# ceph log last cephadm
```

这些事件也记录到监控器主机上的 ceph.cephadm.log 文件，以及 monitor 守护进程的 stderr

14.2. CEPH 守护进程日志

您可以通过 stderr 或文件查看 Ceph 守护进程日志。

将日志输出到 stdout

传统上，Ceph 守护进程已记录到 /var/log/ceph。默认情况下，Cephadm 守护进程将日志记录到 stderr，日志由容器运行时环境捕获。对于大多数系统，默认情况下，这些日志发送到 journald，并可通过 journalctl 命令访问。

- 例如，要在 host01 上查看存储集群 ID 为 5c5a50ae-272a-455d-99e9-32c6a013e694 的守护进程的日志：

  示例

```
[ceph: root@host01 /]# journalctl -u ceph-5c5a50ae-272a-455d-99e9-
32c6a013e694@host01
```

这在日志级别较低时，这适用于正常的 Cephadm 操作。

- 要禁用日志记录到 stderr，请设置以下值：

  示例

```
[ceph: root@host01 /]# ceph config set global log_to_stderr false
[ceph: root@host01 /]# ceph config set global mon_cluster_log_to_stderr false
```

登录到文件

您也可以将 Ceph 守护进程配置为记录到文件，而非 stderr。在登录到文件时，Ceph 日志位于 /var/log/ceph/*CLUSTER_FSID* 中。

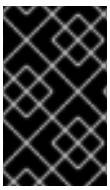
- 要启用日志记录到文件，设置以下值：

示例

```
[ceph: root@host01 /]# ceph config set global log_to_file true
[ceph: root@host01 /]# ceph config set global mon_cluster_log_to_file true
```

注意

红帽建议禁用日志记录到 stderr，以避免重复日志。



重要

目前还不支持将日志轮转到非默认路径。

默认情况下，Cephadm 在每一主机上设置日志轮转，以轮换这些文件。您可以通过修改 /etc/logrotate.d/ceph.*CLUSTER_FSID* 来配置日志记录保留调度。

14.3. 数据位置

Cephadm 守护进程数据和日志位于与较旧版本的 Ceph 相比稍有不同的位置：

-

/var/log/ceph/*CLUSTER_FSID* 包含所有存储集群日志。请注意，默认情况下，Cephadm 日志会导向 stderr 和容器运行时，因此通常不会保存这些日志。

- /var/lib/ceph/*CLUSTER_FSID* 包含所有集群守护进程数据，除了日志之外。
- var/lib/ceph/*CLUSTER_FSID*/*DAEMON_NAME* 包含特定守护进程的所有数据。
- /var/lib/ceph/*CLUSTER_FSID*/crash 包含存储集群的崩溃报告。
- /var/lib/ceph/*CLUSTER_FSID*/removed 包含有状态守护进程的旧守护进程数据目录，如 monitor 或 Prometheus，已被 Cephadm 删除。

磁盘用量

一些 Ceph 守护进程可能会将大量数据存储在 /var/lib/ceph 中，特别是 monitor 和 Prometheus 守护进程，因此红帽建议将该目录移到自己的磁盘、分区或逻辑卷中，以便 root 文件系统不会被填充。

14.4. CEPHADM 自定义配置文件

Cephadm 支持为守护进程指定各种配置文件。您必须提供配置文件的内容和应挂载它的守护进程容器中的位置。

YAML spec 由指定的自定义配置文件应用。Cephadm 重新部署指定配置文件的守护进程。然后，这些文件会在守护进程的容器中挂载到指定位置。

- 您可以使用自定义配置文件应用 YAML 规格：

示例

```
service_type: grafana
service_name: grafana
custom_configs:
  - mount_path: /etc/example.conf
    content: |
      setting1 = value1
      setting2 = value2
```

```
  - mount_path: /usr/share/grafana/example.cert
    content: |
-----BEGIN PRIVATE KEY-----
V2VyIGRhcyBsaWVzdCBpc3QgZG9vZi4gTG9yZW0gaXBzdW0gZG9sb3Igc2l0IGFtZXQsIGNv
bnNldGV0dXIgc2FkaXBzY2luZyBlbGl0ciwgc2VkIGRpYW0gbm9udW15IGVpcm1vZCB0ZW1w
b3IgaW52aWR1bnQgdXQgbGFib3JlIGV0IGRvbG9yZSBtYWduYSBhbGlxdXlhbSBlcmF0LCBz
ZWQgZGlhbSB2b2x1cHR1YS4gQXQgdmVybyBlb3MgZXQgYWNjdXNhbSBldCBqdXN0byBkdW
W8=
-----END PRIVATE KEY-----
-----BEGIN CERTIFICATE-----
V2VyIGRhcyBsaWVzdCBpc3QgZG9vZi4gTG9yZW0gaXBzdW0gZG9sb3Igc2l0IGFtZXQsIGNv
bnNldGV0dXIgc2FkaXBzY2luZyBlbGl0ciwgc2VkIGRpYW0gbm9udW15IGVpcm1vZCB0ZW1w
b3IgaW52aWR1bnQgdXQgbGFib3JlIGV0IGRvbG9yZSBtYWduYSBhbGlxdXlhbSBlcmF0LCBz
ZWQgZGlhbSB2b2x1cHR1YS4gQXQgdmVybyBlb3MgZXQgYWNjdXNhbSBldCBqdXN0byBkdW
W8=
-----END CERTIFICATE-----
```

- **您可以为守护进程在容器中挂载新配置文件：**

语法

```
ceph orch redeploy SERVICE_NAME
```

示例

```
[ceph: root@host01 /]# ceph orch redeploy grafana
```

第 15 章 CEPHADM 健康检查

作为存储管理员，您可以使用 Cephadm 模块提供的附加健康检查来监控 Red Hat Ceph Storage 集群。这是存储集群提供的默认健康检查补充。

15.1. CEPHADM 操作健康检查

当 Cephadm 模块激活时，将执行健康检查。您可以收到以下健康警告：

CEPHADM_PAUSED

Cephadm 后台工作可以使用 ceph orch pause 命令暂停。Cephadm 继续执行被动监控活动，如检查主机和守护进程状态，但它不会像部署或移除守护进程一样进行任何更改。您可以使用 ceph orch resume 命令恢复 Cephadm 工作。

CEPHADM_STRAY_HOST

一个或多个主机正在运行 Ceph 守护进程，但未注册为由 Cephadm 模块管理的主机。这意味着这些服务目前不由 Cephadm 管理，例如，ceph orch ps 命令中包含的重启和升级。您可以使用 ceph orch host add *HOST_NAME* 命令来管理主机，但请确保已配置了对远程主机的 SSH 访问。或者，您可以手动连接到主机，并确保该主机上的服务被删除或迁移到由 Cephadm 管理的主机。您还可以通过设置 ceph config set mgr mgr/cephadm/warn_on_stray_hosts false来禁用此警告

CEPHADM_STRAY_DAEMON

一个或多个 Ceph 守护进程正在运行，但不由 Cephadm 模块管理。这可能是因为使用其他工具部署，或者因为手动启动它们。这些服务目前不由 Cephadm 管理，例如，ceph orch ps 命令中包含的重启和升级。

如果守护进程是 monitor 或 OSD 守护进程的有状态，则 Cephadm 应该采用这些守护进程。对于无状态守护进程，您可以使用 ceph orch apply 命令置备新的守护进程，然后停止非受管守护进程。

您可以通过设置 ceph config set mgr/cephadm/warn_on_stray_daemons false 来禁用此运行状况警告。

CEPHADM_HOST_CHECK_FAILED

对一个或多个主机的基本 Cephadm 主机检查（验证 that:name: 值）失败。

- 主机可以访问，您可以执行 Cephadm。
- 主机满足基本先决条件，如作为 Podman 的工作容器运行时和工作时间同步。如果测试失败，Cephadm 将无法管理该主机上的服务。

您可以使用 ceph cephadm check-host *HOST_NAME* 命令手动运行此检查。您可以使用 ceph orch host rm *HOST_NAME* 命令从管理中删除损坏的主机。您可以通过设置 ceph config set mgr mgr/cephadm/warn_on_failed_host_check false 来禁用此健康警告。

15.2. CEPHADM 配置健康检查

Cephadm 定期扫描存储集群中的每个主机，以了解操作系统、磁盘和 NIC 的状态。这些事实分析为存储集群中主机的一致性，以识别任何配置。配置检查是一个可选功能。

- 您可以使用以下命令启用此功能：

  示例

```
[ceph: root@host01 /]# ceph config set mgr mgr/cephadm/config_checks_enabled true
```

配置检查会在每个主机扫描后触发，这是一分钟的持续时间。

- ceph -W cephadm 命令显示配置检查的当前状态和结果，如下所示：

  禁用状态

  示例

```
ALL cephadm checks are disabled, use 'ceph config set mgr
mgr/cephadm/config_checks_enabled true' to enable
```

启用状态

示例

```
CEPHADM 8/8 checks enabled and executed (0 bypassed, 0 disabled). No issues detected
```

配置检查本身通过多个 **cephadm** 子命令进行管理。

- 要确定配置检查是否已启用，请运行以下命令：

示例

```
[ceph: root@host01 /]# ceph cephadm config-check status
```

此命令将配置检查器的状态返回为 **Enabled** 或 **Disabled**。

- 要列出所有配置检查及其状态，请运行以下命令：

示例

```
[ceph: root@host01 /]# ceph cephadm config-check ls
NAME             HEALTHCHECK                      STATUS   DESCRIPTION
kernel_security  CEPHADM_CHECK_KERNEL_LSM          enabled  checks
SELINUX/Apparmor profiles are consistent across cluster hosts
os_subscription  CEPHADM_CHECK_SUBSCRIPTION       enabled  checks subscription
```

```
states are consistent for all cluster hosts
public_network   CEPHADM_CHECK_PUBLIC_MEMBERSHIP  enabled  check that all hosts
have a NIC on the Ceph public_netork
osd_mtu_size     CEPHADM_CHECK_MTU               enabled  check that OSD hosts share a
common MTU setting
osd_linkspeed    CEPHADM_CHECK_LINKSPEED         enabled  check that OSD hosts
share a common linkspeed
network_missing  CEPHADM_CHECK_NETWORK_MISSING   enabled  checks that the
cluster/public networks defined exist on the Ceph hosts
ceph_release     CEPHADM_CHECK_CEPH_RELEASE      enabled  check for Ceph version
consistency - ceph daemons should be on the same release (unless upgrade is active)
kernel_version   CEPHADM_CHECK_KERNEL_VERSION    enabled  checks that the
MAJ.MIN of the kernel on Ceph hosts is consistent
```

每个配置检查都如下所述：

CEPHADM_CHECK_KERNEL_LSM

存储集群中的每个主机都应该在相同的 Linux 安全模块 (LSM) 状态中运行。例如，如果大多数主机以 enforcing 模式使用 SELINUX 运行，则任何没有在这个模式下运行的主机都将标记为 anomaly，并且会引发警告状态的健康检查。

CEPHADM_CHECK_SUBSCRIPTION

此检查与供应商订阅的状态相关。此检查只针对使用 Red Hat Enterprise Linux 的主机执行，但有助于确认通过有效订阅涵盖所有主机，以便补丁和更新可用。

CEPHADM_CHECK_PUBLIC_MEMBERSHIP

集群的所有成员都应该在至少一个公共网络子网上配置了 NIC。没有处于公共网络上的主机将会依赖于路由，这可能会影响性能。

CEPHADM_CHECK_MTU

OSD 上 NIC 的最大传输单元(MTU)可以是一致性能的关键因素。此检查会检查正在运行 OSD 服务的主机，以确保在集群中 MTU 的配置是一致的。这通过建立大多数主机正在使用的 MTU 设置来确定，任何异常情况会导致 Ceph 健康检查。

CEPHADM_CHECK_LINKSPEED

与 MTU 检查类似，链路速度一致性也是集群性能的一个因素。此检查决定了大多数 OSD 主机共享的链路速度，从而对以较低链接速度设置的主机进行健康检查。

CEPHADM_CHECK_NETWORK_MISSING

public_network 和 cluster_network 设置支持 IPv4 和 IPv6 的子网定义。如果在存储集群的任何主机上找不到这些设置，则会引发健康检查。

CEPHADM_CHECK_CEPH_RELEASE

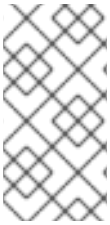
在正常操作下，Ceph 集群应在相同的 Ceph 发行版本中运行守护进程，如所有 Red Hat Ceph Storage 集群 5 版本。此检查将查看每个守护进程的活动发行版本，并报告任何异常情况作为健康检查。如果升级过程在集群内处于活跃状态，则会绕过这个检查。

CEPHADM_CHECK_KERNEL_VERSION

检查 OS 内核版本以获得主机之间的一致性。再次使用大多数主机来识别异常情况。

第 16 章 使用 CEPHADM-ANSIBLE 模块管理红帽 CEPH 存储集群

作为存储管理员，您可以在 Ansible playbook 中使用 cephadm-ansible 模块来管理 Red Hat Ceph Storage 集群。cephadm-ansible 软件包提供了多个模块，可以嵌套 cephadm 调用，以让您编写自己的唯一 Ansible playbook 来管理集群。



注意

目前，cephadm-ansible 模块仅支持最重要的任务。并非 cephadm-ansible 模块涵盖的任何操作都必须在 playbook 中使用 command 或 shell Ansible 模块来完成。

16.1. CEPHADM-ANSIBLE 模块

cephadm-ansible 模块是一组模块，通过打包 cephadm 和 ceph orch 命令提供一个打包程序来简化 Ansible playbook 的编写过程。您可以使用模块自行编写 Ansible playbook，以通过一个或多个模块来管理集群。

cephadm-ansible 软件包包含以下模块：

- cephadm_bootstrap
- ceph_orch_host
- ceph_config
- ceph_orch_apply
- ceph_orch_daemon
- cephadm_registry_login

16.2. CEPHADM-ANSIBLE 模块选项

下表列出了 cephadm-ansible 模块的可用选项。使用 Ansible playbook 中的模块时，需要设置列为必需选项。以默认值 true 列出的选项表示在使用模块时会自动设置该选项，且不需要在 playbook 中指定它。例如，对于 cephadm_bootstrap 模块，将安装 Ceph 仪表板，除非设置了 dashboard: false。

表 16.1. cephadm_bootstrap 模块的可用选项。

cephadm_bootstrap	描述	必填	默认
mon_ip	Ceph 监控 IP 地址。	true	
image	Ceph 容器镜像。	false	
docker	使用 **docker** 而不是 **podman**。	false	
fsid	定义 Ceph FSID。	false	
pull	拉取 Ceph 容器镜像。	false	true
dashboard	部署 Ceph 仪表板。	false	true
dashboard_user	指定特定的 Ceph Dashboard 用户。	false	
dashboard_password	Ceph 仪表板密码。	false	
monitoring	部署监控堆栈。	false	true
firewalld	使用 firewalld 管理防火墙规则。	false	true
allow_overwrite	允许覆盖现有 --output-config、--output-keyring 或 --output-pub-ssh-key 文件。	false	false
registry_url	自定义 registry 的 URL。	false	
registry_username	自定义 registry 的用户名。	false	
registry_password	自定义 registry 密码。	false	
registry_json	带有自定义 registry 登录信息的 JSON 文件。	false	

cephadm_bootstrap	描述	必填	默认
ssh_user	用于 **cephadm** ssh 到主机的 SSH 用户。	false	
ssh_config	用于 **cephadm** SSH 客户端的 SSH 配置文件路径。	false	
allow_fqdn_hostname	允许主机名，即完全限定域名(FQDN)。	false	false
cluster_network	用于集群复制、恢复和心跳的子网。	false	

表 16.2. ceph_orch_host 模块的可用选项。

ceph_orch_host	描述	必填	默认
fsid	要与之交互的 Ceph 集群的 FSID。	false	
image	要使用的 Ceph 容器镜像。	false	
name	要添加的、删除或更新的主机的名称。	true	
address	主机的 IP 地址。	当 **state** 为 **present** 时为 true。	
set_admin_label	在指定主机上设置 **_admin** 标签。	false	false
labels	应用到主机的标签列表。	false	[]
state	如果设置为 **present**，它将确保名称中指定的 **名称** 存在。如果设置为 **absent**，它将删除 **名称** 中指定的主机。如果设置为 **drain**，它将调度从 **名称** 中指定的主机中删除所有守护进程。	false	存在

表 16.3. ceph_config 模块的可用选项

ceph_config	描述	必填	默认
fsid	要与之交互的 Ceph 集群的 FSID。	false	
image	要使用的 Ceph 容器镜像。	false	
action	在 **option** 中指定的参数为 **set** 或 **get**。	false	set
who	哪个守护进程将配置设置为。	true	
选项	要进行 **set** 或 **get** 的参数名称。	true	
value	要设置的参数值。	如果操作是 **set** 为 true	

表 16.4. ceph_orch_apply 模块的可用选项。

ceph_orch_apply	描述	必填
fsid	要与之交互的 Ceph 集群的 FSID。	false
image	要使用的 Ceph 容器镜像。	false
spec	要应用的服务规格。	true

表 16.5. ceph_orch_daemon 模块的可用选项。

ceph_orch_daemon	描述	必填
fsid	要与之交互的 Ceph 集群的 FSID。	false
image	要使用的 Ceph 容器镜像。	false
state	在 **name** 中指定的理想的服务状态。	true 如果为 **started**，它将确保服务已启动。 如果为 **stopped**，它将确保该服务已经停止。 如果为 **restarted**，它将重启该服务。

ceph_orch_daemon	描述	必填
daemon_id	服务的 ID。	true
daemon_type	服务的类型。	true

表 16.6. cephadm_registry_login 模块的可用选项

cephadm_registry_login	描述	必填	默认
state	登录或注销 registry。	false	login
docker	使用 **docker** 而不是 **podman**。	false	
registry_url	自定义 registry 的 URL。	false	
registry_username	自定义 registry 的用户名。	当 **state** 为 **login** 时为 **true**	
registry_password	自定义 registry 密码。	当 **state** 为 **login** 时为 **true**	
registry_json	到一个 JSON 文件的路径。在运行此任务前，该文件必须存在于远程主机上。目前不支持这个选项。		

16.3. 使用 CEPHADM_BOOTSTRAP 和 CEPHADM_REGISTRY_LOGIN 模块引导存储集群

作为存储管理员，您可以使用 Ansible 中的 cephadm_bootstrap 和 cephadm_registry_login 模块来引导存储集群。

先决条件

- **第一个 Ceph 监控容器的 IP 地址，也是存储集群中第一个节点的 IP 地址。**
- **登录到 registry.redhat.io。**
- **至少 10 GB 的可用空间用于 /var/lib/containers/。**

- Red Hat Enterprise Linux 8.10 或 9.4 或更高版本，将 ansible-core 捆绑到 AppStream 中。
- 在 Ansible 管理节点上安装 cephadm-ansible 软件包。
- 在存储集群中的所有主机上设置免密码 SSH。
- 主机通过 CDN 注册。

流程

1. 登录 Ansible 管理节点。
2. 进入 Ansible 管理节点上的 /usr/share/cephadm-ansible 目录：

示例

```
[ceph-admin@admin ~]$ cd /usr/share/cephadm-ansible
```

3. 创建 hosts 文件并添加主机、标签和监控存储集群中第一个主机的 IP 地址：

语法

```
sudo vi INVENTORY_FILE

HOST1 labels="['LABEL1', 'LABEL2']"
HOST2 labels="['LABEL1', 'LABEL2']"
HOST3 labels="['LABEL1']"

[admin]
ADMIN_HOST monitor_address=MONITOR_IP_ADDRESS labels="['ADMIN_LABEL',
'LABEL1', 'LABEL2']"
```

示例

```
[ceph-admin@admin cephadm-ansible]$ sudo vi hosts

host02 labels="['mon', 'mgr']"
host03 labels="['mon', 'mgr']"
host04 labels="['osd']"
host05 labels="['osd']"
host06 labels="['osd']"

[admin]
host01 monitor_address=10.10.128.68 labels="['_admin', 'mon', 'mgr']"
```

4. 运行 **preflight playbook：**

语法

```
ansible-playbook -i INVENTORY_FILE cephadm-preflight.yml --extra-vars "ceph_origin=rhcs"
```

示例

```
[ceph-admin@admin cephadm-ansible]$ ansible-playbook -i hosts cephadm-preflight.yml --
extra-vars "ceph_origin=rhcs"
```

5. 创建 **playbook** 以启动集群：

语法

```
sudo vi PLAYBOOK_FILENAME.yml

---
- name: NAME_OF_PLAY
  hosts: BOOTSTRAP_HOST
  become: USE_ELEVATED_PRIVILEGES
  gather_facts: GATHER_FACTS_ABOUT_REMOTE_HOSTS
  tasks:
    -name: NAME_OF_TASK
     cephadm_registry_login:
       state: STATE
       registry_url: REGISTRY_URL
       registry_username: REGISTRY_USER_NAME
       registry_password: REGISTRY_PASSWORD

    - name: NAME_OF_TASK
      cephadm_bootstrap:
        mon_ip: "{{ monitor_address }}"
        dashboard_user: DASHBOARD_USER
        dashboard_password: DASHBOARD_PASSWORD
        allow_fqdn_hostname: ALLOW_FQDN_HOSTNAME
        cluster_network: NETWORK_CIDR
```

示例

```
[ceph-admin@admin cephadm-ansible]$ sudo vi bootstrap.yml

---
- name: bootstrap the cluster
  hosts: host01
  become: true
  gather_facts: false
  tasks:
    - name: login to registry
      cephadm_registry_login:
        state: login
        registry_url: registry.redhat.io
        registry_username: user1
        registry_password: mypassword1

    - name: bootstrap initial cluster
      cephadm_bootstrap:
        mon_ip: "{{ monitor_address }}"
        dashboard_user: mydashboarduser
```

```
dashboard_password: mydashboardpassword
allow_fqdn_hostname: true
cluster_network: 10.10.128.0/28
```

6. 运行 **playbook：**

语法

```
ansible-playbook -i INVENTORY_FILE PLAYBOOK_FILENAME.yml -vvv
```

示例

```
[ceph-admin@admin cephadm-ansible]$ ansible-playbook -i hosts bootstrap.yml -vvv
```

验证

- 在运行 **playbook** 后检查 **Ansible** 输出。

16.4. 使用 CEPH_ORCH_HOST 模块添加或删除主机

作为存储管理员，您可以使用 **Ansible playbook** 中的 **ceph_orch_host** 模块添加和删除存储集群中的主机。

先决条件

- 一个正在运行的 **Red Hat Ceph Storage** 集群。

- 将节点注册到 CDN 并附加订阅。
- 具有 sudo 的 Ansible 用户，对存储集群中的所有节点进行免密码 SSH 访问。
- 在 Ansible 管理节点上安装 cephadm-ansible 软件包。
- 新主机具有存储集群的公共 SSH 密钥。有关将存储集群的公共 SSH 密钥复制到新主机的更多信息，请参阅 *Red Hat Ceph Storage 安装指南中的添加主机。* *https://access.redhat.com/documentation/zh-cn/red_hat_ceph_storage/7/html-single/installation_guide/#adding-hosts_install*

流程

1. *使用以下步骤在集群中添加新主机：*

   a. *登录 Ansible 管理节点。*

   b. *进入 Ansible 管理节点上的 /usr/share/cephadm-ansible 目录：*

   示例

   ```
   [ceph-admin@admin ~]$ cd /usr/share/cephadm-ansible
   ```

   c. *将新主机和标签添加到 Ansible 清单文件。*

   语法

   ```
   sudo vi INVENTORY_FILE

   NEW_HOST1 labels="['LABEL1', 'LABEL2']"
   NEW_HOST2 labels="['LABEL1', 'LABEL2']"
   ```

```
NEW_HOST3 labels="['LABEL1']"

[admin]
ADMIN_HOST monitor_address=MONITOR_IP_ADDRESS labels="['ADMIN_LABEL',
'LABEL1', 'LABEL2']"
```

示例

```
[ceph-admin@admin cephadm-ansible]$ sudo vi hosts

host02 labels="['mon', 'mgr']"
host03 labels="['mon', 'mgr']"
host04 labels="['osd']"
host05 labels="['osd']"
host06 labels="['osd']"

[admin]
host01 monitor_address= 10.10.128.68 labels="['_admin', 'mon', 'mgr']"
```

注意

如果您之前已将新主机添加到 Ansible 清单文件，并在主机上运行 preflight playbook，请跳至第 3 步。

d. **使用 --limit 选项运行 preflight playbook：**

语法

```
ansible-playbook -i INVENTORY_FILE cephadm-preflight.yml --extra-vars
"ceph_origin=rhcs" --limit NEWHOST
```

示例

```
[ceph-admin@admin cephadm-ansible]$ ansible-playbook -i hosts cephadm-preflight.yml
--extra-vars "ceph_origin=rhcs" --limit host02
```

preflight playbook 在新主机上安装 podman、lvm2、chronyd 和 cephadm。安装完成后，cephadm 驻留在 /usr/sbin/ 目录中。

e.

创建 **playbook** 以将新主机添加到集群中：

语法

```
sudo vi PLAYBOOK_FILENAME.yml

---
- name: PLAY_NAME
  hosts: HOSTS_OR_HOST_GROUPS
  become: USE_ELEVATED_PRIVILEGES
  gather_facts: GATHER_FACTS_ABOUT_REMOTE_HOSTS
  tasks:
    - name: NAME_OF_TASK
      ceph_orch_host:
        name: "{{ ansible_facts['hostname'] }}"
        address: "{{ ansible_facts['default_ipv4']['address'] }}"
        labels: "{{ labels }}"
      delegate_to: HOST_TO_DELEGATE_TASK_TO

    - name: NAME_OF_TASK
      when: inventory_hostname in groups['admin']
      ansible.builtin.shell:
        cmd: CEPH_COMMAND_TO_RUN
      register: REGISTER_NAME

    - name: NAME_OF_TASK
      when: inventory_hostname in groups['admin']
      debug:
        msg: "{{ REGISTER_NAME.stdout }}"
```

注意

默认情况下，Ansible 在与 playbook 的 hosts 行匹配的主机上执行所有任务。ceph orch 命令必须在包含管理员密钥环和 Ceph 配置文件的主机上运行。使用 delegate_to 关键字指定集群中的 admin 主机。

示例

```
[ceph-admin@admin cephadm-ansible]$ sudo vi add-hosts.yml

---
- name: add additional hosts to the cluster
  hosts: all
  become: true
  gather_facts: true
  tasks:
    - name: add hosts to the cluster
      ceph_orch_host:
        name: "{{ ansible_facts['hostname'] }}"
        address: "{{ ansible_facts['default_ipv4']['address'] }}"
        labels: "{{ labels }}"
      delegate_to: host01

    - name: list hosts in the cluster
      when: inventory_hostname in groups['admin']
      ansible.builtin.shell:
        cmd: ceph orch host ls
      register: host_list

    - name: print current list of hosts
      when: inventory_hostname in groups['admin']
      debug:
        msg: "{{ host_list.stdout }}"
```

在本例中，playbook 将新主机添加到集群中，并显示当前的主机列表。

f. 运行 playbook 以将其他主机添加到集群中：

语法

```
ansible-playbook -i INVENTORY_FILE PLAYBOOK_FILENAME.yml
```

示例

```
[ceph-admin@admin cephadm-ansible]$ ansible-playbook -i hosts add-hosts.yml
```

2. **使用以下步骤从集群中删除主机：**

   a. **登录 Ansible 管理节点。**

   b. **进入 Ansible 管理节点上的 /usr/share/cephadm-ansible 目录：**

   示例

   ```
   [ceph-admin@admin ~]$ cd /usr/share/cephadm-ansible
   ```

   c. **创建 playbook 以从集群中删除主机或主机：**

   语法

   ```
   sudo vi PLAYBOOK_FILENAME.yml

   ---
   - name: NAME_OF_PLAY
     hosts: ADMIN_HOST
     become: USE_ELEVATED_PRIVILEGES
     gather_facts: GATHER_FACTS_ABOUT_REMOTE_HOSTS
   ```

```
tasks:
  - name: NAME_OF_TASK
    ceph_orch_host:
      name: HOST_TO_REMOVE
      state: STATE

  - name: NAME_OF_TASK
    ceph_orch_host:
      name: HOST_TO_REMOVE
      state: STATE
    retries: NUMBER_OF_RETRIES
    delay: DELAY
    until: CONTINUE_UNTIL
    register: REGISTER_NAME

  - name: NAME_OF_TASK
    ansible.builtin.shell:
      cmd: ceph orch host ls
    register: REGISTER_NAME

  - name: NAME_OF_TASK
      debug:
        msg: "{{ REGISTER_NAME.stdout }}"
```

示例

```
[ceph-admin@admin cephadm-ansible]$ sudo vi remove-hosts.yml

---
- name: remove host
  hosts: host01
  become: true
  gather_facts: true
  tasks:
    - name: drain host07
      ceph_orch_host:
        name: host07
        state: drain

    - name: remove host from the cluster
      ceph_orch_host:
        name: host07
        state: absent
      retries: 20
      delay: 1
      until: result is succeeded
      register: result

     - name: list hosts in the cluster
```

```
    ansible.builtin.shell:
      cmd: ceph orch host ls
    register: host_list

  - name: print current list of hosts
    debug:
      msg: "{{ host_list.stdout }}"
```

在本例中，playbook 任务排空 host07 上的所有守护进程，从集群中删除主机，并显示当前主机列表。

d. **运行 playbook 以从集群中删除主机：**

语法

```
ansible-playbook -i INVENTORY_FILE PLAYBOOK_FILENAME.yml
```

示例

```
[ceph-admin@admin cephadm-ansible]$ ansible-playbook -i hosts remove-hosts.yml
```

验证

- **查看 Ansible 任务输出显示集群中主机的当前列表：**

示例

```
TASK [print current hosts]
******************************************************************************************************
```

```
Friday 24 June 2022  14:52:40 -0400 (0:00:03.365)       0:02:31.702 ***********
ok: [host01] =>
  msg: |-
    HOST    ADDR          LABELS         STATUS
    host01  10.10.128.68  _admin mon mgr
    host02  10.10.128.69  mon mgr
    host03  10.10.128.70  mon mgr
    host04  10.10.128.71  osd
    host05  10.10.128.72  osd
    host06  10.10.128.73  osd
```

16.5. 使用 CEPH_CONFIG 模块设置配置选项

作为存储管理员，您可以使用 ceph_config 模块设置或获取 Red Hat Ceph Storage 配置选项。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。
- 具有 sudo 的 Ansible 用户，对存储集群中的所有节点进行免密码 SSH 访问。
- 在 Ansible 管理节点上安装 cephadm-ansible 软件包。
- Ansible 清单文件包含集群和 admin 主机。

流程

1. 登录 Ansible 管理节点。
2. 进入 Ansible 管理节点上的 /usr/share/cephadm-ansible 目录：

示例

```
[ceph-admin@admin ~]$ cd /usr/share/cephadm-ansible
```

3. ***使用配置更改创建 playbook：***

语法

```
sudo vi PLAYBOOK_FILENAME.yml

---
- name: PLAY_NAME
  hosts: ADMIN_HOST
  become: USE_ELEVATED_PRIVILEGES
  gather_facts: GATHER_FACTS_ABOUT_REMOTE_HOSTS
  tasks:
    - name: NAME_OF_TASK
      ceph_config:
        action: GET_OR_SET
        who: DAEMON_TO_SET_CONFIGURATION_TO
        option: CEPH_CONFIGURATION_OPTION
        value: VALUE_OF_PARAMETER_TO_SET

    - name: NAME_OF_TASK
      ceph_config:
        action: GET_OR_SET
        who: DAEMON_TO_SET_CONFIGURATION_TO
        option: CEPH_CONFIGURATION_OPTION
      register: REGISTER_NAME

    - name: NAME_OF_TASK
      debug:
        msg: "MESSAGE_TO_DISPLAY {{ REGISTER_NAME.stdout }}"
```

示例

```
[ceph-admin@admin cephadm-ansible]$ sudo vi change_configuration.yml

---
- name: set pool delete
  hosts: host01
  become: true
  gather_facts: false
  tasks:
```

```
- name: set the allow pool delete option
  ceph_config:
    action: set
    who: mon
    option: mon_allow_pool_delete
    value: true

- name: get the allow pool delete setting
  ceph_config:
    action: get
    who: mon
    option: mon_allow_pool_delete
  register: verify_mon_allow_pool_delete

- name: print current mon_allow_pool_delete setting
  debug:
    msg: "the value of 'mon_allow_pool_delete' is {{ verify_mon_allow_pool_delete.stdout }}"
```

在本例中，playbook 首先将 mon_allow_pool_delete 选项设置为 false。然后，playbook 获取当前的 mon_allow_pool_delete 设置，并在 Ansible 输出中显示值。

4. 运行 **playbook：**

语法

```
ansible-playbook -i INVENTORY_FILE _PLAYBOOK_FILENAME.yml
```

示例

```
[ceph-admin@admin cephadm-ansible]$ ansible-playbook -i hosts change_configuration.yml
```

验证

- 检查 playbook 任务的输出。

示例

```
TASK [print current mon_allow_pool_delete setting]
*************************************************************
Wednesday 29 June 2022  13:51:41 -0400 (0:00:05.523)       0:00:17.953 ********
ok: [host01] =>
  msg: the value of 'mon_allow_pool_delete' is true
```

其它资源

- 有关配置选项的更多详细信息，请参阅 Red Hat Ceph Storage 配置指南。

16.6. 使用 CEPH_ORCH_APPLY 模块应用服务规格

作为存储管理员，您可以使用 Ansible playbook 中的 ceph_orch_apply 模块将服务规格应用到存储集群。服务规格是一个数据结构，它指定用于部署 Ceph 服务的服务属性和配置设置。您可以使用服务规格来部署 Ceph 服务类型，如 mon、crash、mds、mgr、osd、rdb 或 rbd-mirror。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。
- 具有 sudo 的 Ansible 用户，对存储集群中的所有节点进行免密码 SSH 访问。
- 在 Ansible 管理节点上安装 cephadm-ansible 软件包。
- Ansible 清单文件包含集群和 admin 主机。

流程

1. *登录 **Ansible** 管理节点。*

2. *进入 **Ansible** 管理节点上的 **/usr/share/cephadm-ansible** 目录：*

示例

```
[ceph-admin@admin ~]$ cd /usr/share/cephadm-ansible
```

3. *使用服务规格创建 **playbook**：*

语法

```
sudo vi PLAYBOOK_FILENAME.yml

---
- name: PLAY_NAME
  hosts: HOSTS_OR_HOST_GROUPS
  become: USE_ELEVATED_PRIVILEGES
  gather_facts: GATHER_FACTS_ABOUT_REMOTE_HOSTS
  tasks:
    - name: NAME_OF_TASK
      ceph_orch_apply:
        spec: |
          service_type: SERVICE_TYPE
          service_id: UNIQUE_NAME_OF_SERVICE
          placement:
            host_pattern: 'HOST_PATTERN_TO_SELECT_HOSTS'
            label: LABEL
          spec:
            SPECIFICATION_OPTIONS:
```

示例

```
[ceph-admin@admin cephadm-ansible]$ sudo vi deploy_osd_service.yml

---
- name: deploy osd service
  hosts: host01
  become: true
  gather_facts: true
  tasks:
    - name: apply osd spec
      ceph_orch_apply:
        spec: |
          service_type: osd
          service_id: osd
          placement:
            host_pattern: '*'
            label: osd
          spec:
            data_devices:
              all: true
```

在本例中，playbook 在所有主机上部署 Ceph OSD 服务，其标签为 osd。

4. *运行 **playbook：***

语法

```
ansible-playbook -i INVENTORY_FILE _PLAYBOOK_FILENAME.yml
```

示例

```
[ceph-admin@admin cephadm-ansible]$ ansible-playbook -i hosts deploy_osd_service.yml
```

验证

- 检查 playbook 任务的输出。

其它资源

- 如需了解有关服务规格选项的更多详细信息，请参阅 Red Hat Ceph Storage Operations Guide。

16.7. 使用 CEPH_ORCH_DAEMON 模块管理 CEPH 守护进程状态

作为存储管理员，您可以使用 Ansible playbook 中的 ceph_orch_daemon 模块在主机上启动、停止和重启 Ceph 守护进程。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。
- 具有 sudo 的 Ansible 用户，对存储集群中的所有节点进行免密码 SSH 访问。
- 在 Ansible 管理节点上安装 cephadm-ansible 软件包。
- Ansible 清单文件包含集群和 admin 主机。

流程

1. 登录 Ansible 管理节点。
2. 进入 Ansible 管理节点上的 /usr/share/cephadm-ansible 目录：

示例

```
[ceph-admin@admin ~]$ cd /usr/share/cephadm-ansible
```

3. *创建带有守护进程状态更改的 playbook：*

语法

```
sudo vi PLAYBOOK_FILENAME.yml

---
- name: PLAY_NAME
  hosts: ADMIN_HOST
  become: USE_ELEVATED_PRIVILEGES
  gather_facts: GATHER_FACTS_ABOUT_REMOTE_HOSTS
  tasks:
    - name: NAME_OF_TASK
      ceph_orch_daemon:
        state: STATE_OF_SERVICE
        daemon_id: DAEMON_ID
        daemon_type: TYPE_OF_SERVICE
```

示例

```
[ceph-admin@admin cephadm-ansible]$ sudo vi restart_services.yml

---
- name: start and stop services
  hosts: host01
  become: true
  gather_facts: false
  tasks:
    - name: start osd.0
      ceph_orch_daemon:
        state: started
        daemon_id: 0
        daemon_type: osd

    - name: stop mon.host02
      ceph_orch_daemon:
        state: stopped
        daemon_id: host02
        daemon_type: mon
```

在本例中，playbook 启动 ID 为 0 的 OSD，并停止 ID 为 host02 的 Ceph Monitor。

4. *运行 playbook：*

语法

```
ansible-playbook -i INVENTORY_FILE _PLAYBOOK_FILENAME.yml
```

示例

```
[ceph-admin@admin cephadm-ansible]$ ansible-playbook -i hosts restart_services.yml
```

验证

- *检查 playbook 任务的输出。*

附录 A. MCLOCK 配置选项

本节包含 mClock 配置选项列表：

osd_mclock_profile

描述

它设置 mClock 配置集类型，用于根据属于不同类的操作（如后台恢复、回填、pg scrub、snap trim、client op 和 pg deletion ）提供服务质量(QoS)。

启用 内置配置集后，较低级别的 mClock 资源控制参数（即保留、权重和限制）以及一些 Ceph 配置参数是透明的。这不适用于 custom 配置集。

类型

字符串

默认

balanced

有效选择

balanced,high_recovery_ops,high_client_ops,custom

osd_mclock_max_capacity_iops_hdd

描述

它设置最大随机写入 IOPS 容量（以 4 KiB 块大小）来考虑每个 OSD 用于轮转介质。启用 dmclock 配置集时，在 QoS 计算。它仅被视为 osd_op_queue = mclock_scheduler

类型

浮点值

默认

315.0

osd_mclock_max_capacity_iops_ssd

描述

它设置一个最大随机写入 IOPS 容量（以 4 KiB 块大小）来考虑每个 OSD 用于固态介质。

类型

浮点值

默认

21500.0

osd_mclock_cost_per_byte_usec_ssd

描述

在启用 dmclock 配置集时，以微秒表示每个 OSD 的 SDD.Contributes in QoS 计算的成本（以微秒为单位）。它仅被视为 osd_op_queue = mclock_scheduler

类型

浮点值

默认

0.011

osd_mclock_max_sequential_bandwidth_hdd

描述

表示底层设备类型为轮转介质的 OSD 的最大连续带宽（以字节为单位）。这被 mclock 调度程序考虑，以派生在 QoS 计算中使用的成本因素。只适用于 osd_op_queue = mclock_scheduler

类型

大小

默认

150_M

osd_mclock_max_sequential_bandwidth_ssd

描述

表示底层设备类型是固态介质的 OSD 的最大连续带宽（以字节为单位）。这被 mclock 调度程序考虑，以派生在 QoS 计算中使用的成本因素。只适用于 osd_op_queue =

mclock_scheduler

类型

大小

默认

1200_M

osd_mclock_force_run_benchmark_on_init

描述

这会强制在 OSD 初始化或引导时运行 OSD 基准。

类型

布尔值

默认

False

另请参阅

osd_mclock_max_capacity_iops_hdd, osd_mclock_max_capacity_iops_ssd

osd_mclock_skip_benchmark

描述

设置此选项会跳过 OSD 初始化或引导时的 OSD 基准。

类型

布尔值

默认

False

另请参阅

osd_mclock_max_capacity_iops_hdd, osd_mclock_max_capacity_iops_ssd

osd_mclock_override_recovery_settings

描述

设置此选项可启用由 osd_recovery_max_active_hdd、osd_recovery_max_active_ssd 和 osd_max_backfills 选项定义的 mClock 调度程序的恢复或回填限制。

类型

布尔值

默认

False

另请参阅

osd_recovery_max_active_hdd, osd_recovery_max_active_ssd, osd_max_backfills

osd_mclock_iops_capacity_threshold_hdd

描述

它表示 阈值 IOPS 容量为 4KiB 块大小，除了忽略 HDD 的 OSD 的 Ceph OSD bench 结果之外。

类型

浮点值

默认

500.0

osd_mclock_iops_capacity_threshold_ssd

描述

它表示阈值 IOPS 容量为 4KiB 块大小，除了忽略 SSD 的 OSD 的 Ceph OSD bench 结果之外。

类型

浮点值

默认

80000.0

osd_mclock_scheduler_client_res

描述

它是为每个客户端保留的默认 I/O 比例。默认值 0 指定最低的保留保留。任何大于 0 且最多 1.0 的值指定为 OSD 最大 IOPS 容量的比例，每个客户端要保留的最小 IO 比例。

类型

浮点值

默认

0

åˆ†é’Ÿ

0

max

1.0

osd_mclock_scheduler_client_wgt

描述

它是保留的每个客户端的默认 I/O 共享。

类型

未签名的整数

默认

1

osd_mclock_scheduler_client_lim

描述

它是每个客户端与保留相关的默认 I/O 限制。默认值 0 指定任何限制强制，这意味着每个客户端都可以使用 OSD 的最大可能 IOPS 容量。任何大于 0 的值，最多 1.0 指定每个客户端在 OSD 最大 IOPS 容量中收到的比例超过保留的最大 IO 限制。

类型

浮点值

默认

0

åˆ†é’Ÿ

0

max

1.0

osd_mclock_scheduler_background_recovery_res

描述

它是为后台恢复保留的默认 I/O 比例。默认值 0 指定最低的保留保留。大于 0 和最多 1.0 的值指定在 OSD 最大 IOPS 容量的一小部分内为后台恢复操作保留的最小 IO 比例。

类型

浮点值

默认

0

åˆ†é’Ÿ

0

max

1.0

osd_mclock_scheduler_background_recovery_wgt

描述

它表示在保留时每个后台恢复的 I/O 共享。

类型

未签名的整数

默认

1

osd_mclock_scheduler_background_recovery_lim

描述

它表示在保留时进行后台恢复的 I/O 限制。默认值 0 指定任何限制强制，这意味着后台恢复操作可以使用 OSD 的最大可能 IOPS 容量。任何大于 0 的值，最多 1.0 指定后台恢复操作在一小部分 OSD 的最大 IOPS 容量上接收的 IO 限值。

类型

浮点值

默认

0

åˆ†é’Ÿ

0

max

1.0

osd_mclock_scheduler_background_best_effort_res

描述

它表示为后台 best_effort 保留的默认 I/O 比例。默认值 0 指定最低的保留保留。任何大于 0 且最多 1.0 的值指定为 OSD 最大 IOPS 容量的几分之一，为后台 best_effort 操作保留的最小 IO 比例。

类型

浮点值

默认

0

åˆ†é’Ÿ

0

max

1.0

osd_mclock_scheduler_background_best_effort_wgt

描述

它指示每个后台 best_effort 的 I/O 共享。

类型

未签名的整数

默认

1

osd_mclock_scheduler_background_best_effort_lim

描述

它表示与保留相关的后台 best_effort 的 I/O 限制。默认值 0 指定任何限制强制，这意味着 background best_effort 操作可以使用 OSD 的最大可能 IOPS 容量。任何大于 0 和最多 1.0 的值，代表后台 best_effort 操作接收的上限是 OSD 最大 IOPS 容量的几分之一。

类型

浮点值

默认

0

åˆ†é’Ÿ

0

max

1.0

其它资源

有关 osd_op_queue 选项的详情，请参阅 Object Storage Daemon (OSD)配置选项。