Red Hat Ceph Storage 7

管理ガイド

Red Hat Ceph Storage の管理

Last Updated: 2024-03-27

Red Hat Ceph Storage 7 管理ガイド

Red Hat Ceph Storage の管理

法律上の通知

概要

本書では、Red Hat Ceph Storage のプロセスの管理、クラスターの状態の監視、ユーザーの管理、およびデーモンの追加および削除方法を説明します。 Red Hat では、コード、ドキュメント、Web プロパティーにおける配慮に欠ける用語の置き換えに取り組んでいます。まずは、マスター (master)、スレーブ (slave)、ブラックリスト (blacklist)、ホワイトリスト (whitelist) の 4 つの用語の置き換えから始めます。この取り組みは膨大な作業を要するため、今後の複数のリリースで段階的に用語の置き換えを実施して参ります。詳細は、弊社の CTO、Chris Wright のメッセージを参照してください。

目次

第1章 CEPH 管理

Red Hat Ceph Storage クラスターは、全 Ceph デプロイメントの基盤となります。Red Hat Ceph Storage クラスターをデプロイしたら、Red Hat Ceph Storage クラスターの正常な状態を維持し、最適に実行するための管理操作を実行できます。

Red Hat Ceph Storage 管理ガイドは、ストレージ管理者が以下のようなタスクを実行するのに役立ちます。

- Red Hat Ceph Storage クラスターの正常性を確認する方法
- Red Hat Ceph Storage クラスターサービスを起動および停止する方法
- 実行中の Red Hat Ceph Storage クラスターから OSD を追加または削除する方法
- Red Hat Ceph Storage クラスターに保管されたオブジェクトへのユーザー認証およびアクセス制御を管理する方法
- Red Hat Ceph Storage クラスターでオーバーライドを使用する方法
- Red Hat Ceph Storage クラスターのパフォーマンスを監視する方法

基本的な Ceph ストレージクラスターは、2 種類のデーモンで設定されます。

- Ceph Object Storage Device (OSD) は、OSD に割り当てられた配置グループ内にオブジェクトとしてデータを格納します。
- Ceph Monitor はクラスターマップのマスターコピーを維持します。

実稼働システムでは、高可用性を実現する Ceph Monitor が 3 つ以上含まれます。通常、許容可能な負荷分散、データのリバランス、およびデータ復旧に備えて最低 50 OSD が含まれます。

第2章 CEPH のプロセス管理について

ストレージ管理者は、Red Hat Ceph Storage クラスター内の種別またはインスタンスごとに、さまざまな Ceph デーモンを操作できます。これらのデーモンを操作すると、必要に応じてすべての Ceph サービスを開始、停止、および再起動することができます。

2.1. CEPH プロセスの管理

Red Hat Ceph Storage では、すべてのプロセス管理は Systemd サービスを介して行われます。Ceph デーモンの **start**、**restart**、および **stop** を行う場合には毎回、デーモンの種別またはデーモンインスタンスを指定する必要があります。

関連情報

- **systemd** の使用の詳細は、systemctl を使用したシステムサービスの管理 を参照してください。

2.2. SYSTEMCTL コマンドを使用したすべての CEPH デーモンの起動、停止、再起動

Ceph デーモンを停止するホストから、すべての Ceph デーモンをルートユーザーとして開始、停止、および再起動できます。

前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。
- ノードへの **root** アクセスを持つ。

手順

1. デーモンを開始、停止、および再起動するホスト上で systemctl サービスを実行して、サービスの **SERVICE_ID** を取得します。

   例

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

2. すべての Ceph デーモンを起動します。

   構文

   ```
   systemctl start SERVICE_ID
   ```

   例

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

3. すべての Ceph デーモンを停止します。

構文

```
systemctl stop SERVICE_ID
```

例

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

4. すべての Ceph デーモンを再起動します。

構文

```
systemctl restart SERVICE_ID
```

例

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

2.3. すべての CEPH サービスの開始、停止、および再起動

Ceph サービスは、同じ Red Hat Ceph Storage クラスターで実行するように設定された、同じタイプの Ceph デーモンの論理グループです。Ceph のオーケストレーションレイヤーにより、ユーザーはこれらのサービスを一元的に管理できるため、同じ論理サービスに属するすべての Ceph デーモンに影響を与える操作を簡単に実行できます。各ホストで実行されている Ceph デーモンは Systemd サービスを通じて管理されます。Ceph サービスを管理するホストから、すべての Ceph サービスを開始、停止、および再起動できます。

重要

特定ホストの特定 Ceph デーモンを開始、停止、または再起動する場合は、SystemD サービスを使用する必要があります。特定のホストで実行されている SystemD サービスのリストを取得するには、ホストに接続し、次のコマンドを実行します。

例

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

出力には、各 Ceph デーモンを管理するために使用できるサービス名のリストが表示されます。

前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。
- ノードへの **root** アクセスを持つ。

手順

1. Cephadm シェルにログインします。

例

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

2. **ceph orch ls** コマンドを実行して、Red Hat Ceph Storage クラスターで設定された Ceph サービスのリストを取得し、特定サービスの ID を取得します。

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. 特定のサービスを開始するには、次のコマンドを実行します。

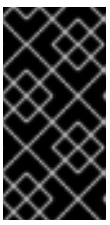
構文

```
ceph orch start SERVICE_ID
```

例

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

4. 特定のサービスを停止するには、次のコマンドを実行します。



重要

ceph orch stop SERVICE_ID コマンドを実行すると、MON および MGR サービスに対してのみ Red Hat Ceph Storage クラスターにアクセスできなくなります。**systemctl stop SERVICE_ID** コマンドを使用して、ホスト内の特定のデーモンを停止することを推奨します。

構文

```
ceph orch stop SERVICE_ID
```

例

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

この例では、**ceph orch stop node-exporter** コマンドは、**node exporter** サービスのすべてのデーモンを削除します。

5. 特定のサービスを再起動するには、次のコマンドを実行します。

構文

```
ceph orch restart SERVICE_ID
```

例

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

2.4. コンテナー内で実行される CEPH デーモンのログファイルの表示

コンテナーホストからの **journald** デーモンを使用して、コンテナーから Ceph デーモンのログファイルを表示します。

前提条件

- Red Hat Ceph Storage ソフトウェアのインストール
- ノードへのルートレベルのアクセス。

手順

1. Ceph ログファイル全体を表示するには、以下の形式で設定される **root** で **journalctl** コマンドを実行します。

構文

```
journalctl -u ceph SERVICE_ID
```

例

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

上記の例では、ID **osd.8** の OSD のログ全体を表示できます。

2. 最近のジャーナルエントリーのみを表示するには、**-f** オプションを使用します。

構文

```
journalctl -fu SERVICE_ID
```

例

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

注記

sosreport ユーティリティーを使用して **journald** ログを表示することもできます。SOS レポートの詳細については、RedHat カスタマーポータルのソリューション sosreport とは何ですか ?Red Hat Enterprise Linux で sosreport を作成する方法は ? を参照してください。

関連情報

- **journalctl** の man ページ

2.5. RED HAT CEPH STORAGE クラスターの電源をオフにして再起動

systemctl コマンドおよび Ceph Orchestrator の 2 つの方法を使用して、Red Hat Ceph Storage クラスターの電源をオフにして再起動できます。クラスターの電源をオフにして再起動する方法のいずれかを選択できます。

2.5.1. systemctl コマンドを使用したクラスターの電源オフおよび再起動

systemctl コマンドアプローチを使用して、Red Hat Ceph Storage クラスターの電源をオフにして再起動できます。このアプローチは、Linux によるサービス停止方法に従います。

前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。
- ルートレベルのアクセス。

手順

Red Hat Ceph Storage クラスターの電源オフ

1. クライアントがこのクラスターおよびその他のクライアント上の Block Device イメージ RADOS Gateway - Ceph Object Gateway を使用しないようにします。
2. Cephadm シェルにログインします。

   例

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

3. 次のステップに進む前に、クラスターの状態が正常な状態 (**Health_OK** およびすべての PG が **active+clean**) である必要があります。Ceph Monitor または OpenStack コントローラーノードなどのクライアントキーリングを持つホストで **ceph status** を実行し、クラスターが正常であることを確認します。

   例

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

4. Ceph File System (**CephFS**) を使用する場合は、**CephFS** クラスターを停止します。

   構文

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

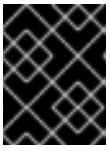
   例

   ```
   [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. **noout** フラグ、**norecover** フラグ、**norebalance** フラグ、**nobackfill** フラグ、**nodown** フラグ、および **pause** フラグを設定します。Ceph Monitor または OpenStack コントローラーなどのクライアントキーリングを持つノードで以下を実行します。

   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

   上記の例は、OSD ノード内のサービスと各 OSD を停止する場合のみであり、各 OSD ノードで繰り返す必要があります。

6. MDS および Ceph Object Gateway ノードがそれぞれ専用のノード上にある場合は、それらの電源をオフにします。

7. OSD ノードを 1 つずつシャットダウンします。

   例

   ```
   [root@host01 ~]# systemctl stop ceph-499829b4-832f-11eb-8d6d-
   001a4a000635@osd.2.service
   ```

8. 監視ノードを 1 つずつシャットダウンします。

   例

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

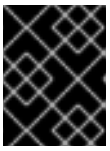
9. 管理ノードをシャットダウンします。

Red Hat Ceph Storage クラスターのリブート

1. ネットワーク機器を使用した場合、Ceph ホストまたはノードの電源を入れる前に、ネットワーク機器の電源が入り、安定していることを確認します。
2. 管理ノードの電源をオンにします。
3. モニターノードの電源をオンにします。

   例

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

   重要

   上記の例は、OSD ノード内のサービスと各 OSD を停止する場合のみであり、各 OSD ノードで繰り返す必要があります。

4. OSD ノードの電源をオンにします。

   例

   ```
   [root@host01 ~]# systemctl start ceph-499829b4-832f-11eb-8d6d-001a4a000635@osd.2.service
   ```

5. すべてのノードが起動するのを待ちます。すべてのサービスが稼働中であり、ノード間の接続に問題がないことを確認します。
6. **noout** フラグ、**norecover** フラグ、**norebalance** フラグ、**nobackfill** フラグ、**nodown** フラグ、および **pause** フラグの設定を解除します。Ceph Monitor または OpenStack コントローラーなどのクライアントキーリングを持つノードで以下を実行します。

   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. Ceph File System (**CephFS**) を使用する場合は、**joinable** フラグを **true** に設定して、**CephFS** クラスターをバックアップします。

   構文

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

   例

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

検証

- クラスターの状態が正常であることを確認します (**Health_OK**、およびすべての PG が **active+clean**)。Ceph Monitor や OpenStack コントローラー **ceph status** ノードなどのクライアントキーリングを持つノードで実行し、クラスターが正常であることを確認します。

例

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

関連情報

- Ceph のインストールの詳細は、Red Hat Ceph Storage インストールガイドを参照してください。

2.5.2. Ceph Orchestrator を使用したクラスターの電源オフおよび再起動

Ceph Orchestrator の機能を使用して、Red Hat Ceph Storage クラスターの電源をオフにして再起動することもできます。ほとんどの場合、システムのログイン 1 回で、クラスターの電源をオフできます。

Ceph Orchestrator は、**start**、**stop**、および **restart** などの複数の操作をサポートします。クラスターの電源をオフにしたり、再起動したりする場合など、**systemctl** でこれらのコマンドを使用できます。

前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。
- ノードへのルートレベルのアクセス。

手順

Red Hat Ceph Storage クラスターの電源オフ

1. このクラスターおよび他のクラスターにある、ユーザーの Block Device イメージや、Ceph Object Gateway をクライアントが使用できないようにします。
2. Cephadm シェルにログインします。

   例

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

3. 次のステップに進む前に、クラスターの状態が正常な状態 (**Health_OK** およびすべての PG が **active+clean**) である必要があります。Ceph Monitor または OpenStack コントローラーノードなどのクライアントキーリングを持つホストで **ceph status** を実行し、クラスターが正常であることを確認します。

   例

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

4. Ceph File System (**CephFS**) を使用する場合は、**CephFS** クラスターを停止します。

   構文

   ```
   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
   ```

   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. **noout** フラグ、**norecover** フラグ、**norebalance** フラグ、**nobackfill** フラグ、**nodown** フラグ、および **pause** フラグを設定します。Ceph Monitor または OpenStack コントローラーなどのクライアントキーリングを持つノードで以下を実行します。

   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. MDS サービスを停止します。

   a. MDS サービス名を取得します。

      例

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

   b. 直前の手順でフェッチされた名前を使用して MDS サービスを停止します。

      構文

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

7. Ceph Object Gateway サービスを停止します。デプロイされたサービスごとに繰り返します。

   a. Ceph Object Gateway サービス名を取得します。

      例

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

b. フェッチされた名前を使用して Ceph Object Gateway サービスを停止します。

構文

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

8. Alertmanager サービスを停止します。

例

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

9. モニタリングスタックの一部である node-exporter サービスを停止します。

例

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

10. Prometheus サービスを停止します。

例

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

11. Grafana ダッシュボードサービスを停止します。

例

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

12. crash サービスを停止します。

例

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

13. cephadm ノードから OSD ノードを 1 つずつシャットダウンします。クラスター内のすべての OSD に対してこの手順を繰り返します。

a. OSD ID を取得します。

例

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

b. フェッチした OSD ID を使用して OSD ノードをシャットダウンします。

例

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

14. モニターを 1 つずつ停止します。

    a. モニターをホストしているホストを特定します。

    例

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

    b. 各ホストでモニターを停止します。

        i. **systemctl** ユニット名を特定します。

        例

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

        ii. サービスを停止します。

        構文

        ```
        systemct stop SERVICE-NAME
        ```

15. すべてのホストをシャットダウンします。

Red Hat Ceph Storage クラスターのリブート

1. ネットワーク機器を使用した場合、Ceph ホストまたはノードの電源を入れる前に、ネットワーク機器の電源が入り、安定していることを確認します。
2. すべての Ceph ホストの電源をオンにします。
3. Cephadm シェルから管理ノードにログインします。

    例

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

4. すべてのサービスが稼働状態にあることを確認します。

    例

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

5. クラスターの正常性が `Health_OK` のステータスであることを確認します。

    例

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

6. **noout** フラグ、**norecover** フラグ、**norebalance** フラグ、**nobackfill** フラグ、**nodown** フラグ、および **pause** フラグの設定を解除します。Ceph Monitor または OpenStack コントローラーなどのクライアントキーリングを持つノードで以下を実行します。

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. Ceph File System (**CephFS**) を使用する場合は、**joinable** フラグを **true** に設定して、**CephFS** クラスターをバックアップします。

構文

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

例

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

検証

- クラスターの状態が正常であることを確認します (**Health_OK**、およびすべての PG が **active+clean**)。Ceph Monitor や OpenStack コントローラー **ceph status** ノードなどのクライアントキーリングを持つノードで実行し、クラスターが正常であることを確認します。

例

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

関連情報

- Ceph のインストールの詳細は、Red Hat Ceph Storage インストールガイドを参照してください。

第3章 CEPH STORAGE クラスターのモニタリング

ストレージ管理者は、Ceph の個々のコンポーネントの正常性を監視すると共に、Red Hat Ceph Storage クラスターの全体的な健全性を監視することができます。

Red Hat Ceph Storage クラスターを稼働したら、ストレージクラスターの監視を開始して、Ceph Monitor デーモンおよび Ceph OSD デーモンが高レベルで実行されていることを確認することができます。Ceph Storage クラスタークライアントは Ceph Monitor に接続して、最新バージョンのストレージクラスターマップを受け取ってから、ストレージクラスター内の Ceph プールへのデータの読み取りおよび書き込みを実施することができます。そのため、モニタークラスターには、Ceph クライアントがデータの読み取りおよび書き込みが可能になる前に、クラスターの状態に関する合意が必要です。

Ceph OSD は、セカンダリー OSD の配置グループのコピーと、プライマリー OSD 上の配置グループをピアにする必要があります。障害が発生した場合、ピアリングは **active + clean** 状態以外のものを反映します。

3.1. CEPH STORAGE クラスターのハイレベル監視

ストレージ管理者は、Ceph デーモンの正常性を監視し、それらが稼働していることを確認します。また、高レベルのモニタリングには、ストレージクラスター容量を確認して、ストレージクラスターが **完全な比率** を超えないようにします。Red Hat Ceph Storage ダッシュボード は、高レベルのモニタリングを実行する最も一般的な方法です。ただし、コマンドラインインターフェイス、Ceph 管理ソケットまたは Ceph API を使用してストレージクラスターを監視することもできます。

3.1.1. ストレージクラスターの正常性の確認

Ceph Storage クラスターを起動してからデータの読み取りまたは書き込みを開始する前に、ストレージクラスターの正常性を確認します。

前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。
- ノードへのルートレベルのアクセス。

手順

1. Cephadm シェルにログインします。

   例

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

2. Ceph Storage クラスターの正常性を確認するには、以下のコマンドを使用します。

   例

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

3. **ceph status** コマンドを実行すると、Ceph ストレージクラスターのステータスを確認できます。

例

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

出力には、次の情報が表示されます。

- Cluster ID
- クラスターの正常性ステータス
- モニターマップエポックおよびモニタークォーラムのステータス
- OSD マップエポックおよび OSD のステータス
- Ceph Manager のステータス
- Object Gateway のステータス
- 配置グループマップのバージョン
- 配置グループとプールの数
- 保存されるデータの想定量および保存されるオブジェクト数
- 保存されるデータの合計量
- IO クライアントの操作。
- クラスターがアップグレード中の場合は、アップグレードプロセスに関する最新情報。Ceph クラスターの起動時に、**HEALTH_WARN XXX num placement groups stale** などの正常性警告が生じる可能性があります。しばらく待ってから再度確認します。ストレージクラスターの準備が整ったら、**ceph health** は **HEALTH_OK** などのメッセージを返すはずです。この時点で、クラスターの使用を開始するのは問題ありません。

3.1.2. ストレージクラスターイベントの監視

コマンドラインインターフェイスを使用して、Ceph Storage クラスターで発生しているイベントを監視することができます。

前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。
- ノードへのルートレベルのアクセス。

手順

1. Cephadm シェルにログインします。

   例

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

2. クラスターの進行中のイベントを監視するには、次のコマンドを実行します。

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 のデータ使用量の計算方法

使用される 値は、使用される生のストレージの **実際** の量を反映します。**xxx GB** / **xxx GB** の値は、クラスターの全体的なストレージ容量のうち、2 つの数字の小さい方の利用可能な量を意味します。概念番号は、複製、クローン、またはスナップショットを作成する前に、保存したデータのサイズを反映します。したがって、Ceph はデータのレプリカを作成し、クローン作成やスナップショットのためにストレージ容量を使用することもあるため、実際に保存されるデータの量は、通常、保存された想定される量を上回ります。

3.1.4. ストレージクラスターの使用統計について

クラスターのデータ使用状況とプール間でのデータ分散を確認するには、**df** オプションを使用します。これは Linux **df** コマンドに似ています。

一部の OSD がクラスターから **OUT として** マークされている場合、すべての OSD が **IN** である場合と比べて、**ceph df** および **ceph status** コマンドの出力の **SIZE/AVAIL/RAW USED** は異なります。**SIZE/AVAIL/RAW USED** は、**IN** 状態にあるすべての OSD の **SIZE** (OSD ディスクサイズ)、**RAW USE** (ディスク上の合計使用スペース)、および **AVAIL** の合計から計算されます。**ceph osd df tree** コマンドの出力で、すべての OSD の **SIZE/AVAIL/RAW USED** の合計を確認できます。

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 コマンドは、quota objects、quota bytes、used compression、および under compression など、その他のプール統計に関する詳細情報を提供します。

出力の **RAW STORAGE** セクションは、ストレージクラスターがデータ用に管理するストレージ容量の概要を説明します。

- **CLASS:** OSD デバイスのクラス。
- **SIZE:** ストレージクラスターが管理するストレージ容量。
  上記の例では、**SIZE** が 90 GiB の場合、レプリケーション係数 (デフォルトでは 3) を考慮しない合計サイズです。レプリケーション係数を考慮した使用可能な合計容量は 90 GiB/3 = 30 GiB です。フル比率 (デフォルトでは 85%) に基づくと、使用可能な最大容量は 30 GiB * 0.85 = 25.5 GiB です。
- **AVAIL:** ストレージクラスターで利用可能な空き容量。
  上記の例では、**SIZE** が 90 GiB、 **USED** 容量が 6 GiB の場合、 **AVAIL** 容量は 84 GiB になります。レプリケーション係数 (デフォルトでは 3) を考慮した使用可能な合計容量は、84 GiB/3 = 28 GiB です。
- **USED:** ユーザーデータが使用する raw ストレージの量。
  上記の例では、100 MiB がレプリケーション係数を考慮した後で利用可能な合計領域です。実際に使用可能なサイズは 33 MiB です。**RAW USED:** ユーザーデータ、内部オーバーヘッド、または予約済み容量で消費される raw ストレージの量。
- **% RAW USED: RAW USED** の割合。この数字は、**full ratio** と **near full ratio** で使用して、ストレージクラスターの容量に達しないようにします。

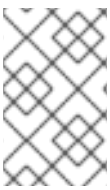
出力の **POOLS** セクションは、プールのリストと、各プールの概念的な使用目的を提供します。このセクションの出力には、レプリカ、クローン、またはスナップショットを **反映しません**。たとえば、1 MB のデータでオブジェクトを保存する場合、概念的な使用量は 1 MB になりますが、実際の使用量は、**size = 3** のクローンやスナップショットなどのレプリカ数によっては 3 MB 以上になる場合があります。

- **POOL:** プールの名前。
- **ID:** プール ID。
- **STOERD:** ユーザーがプールに格納する実際のデータ量。この値は、(k+M)/K 値に基づく生の使用状況データ、オブジェクトのコピー数、プール統計計算時に劣化したオブジェクトの数に基づいて変化します。
- **OBJECTS:** プールごとに保存されるオブジェクトの想定数。これは、**STORED** サイズ * レプリケーション係数です。
- **USED:** メガバイトの場合は **M**、ギガバイトの場合は **G** を付加しない限り、キロバイト単位で保存されたデータの想定量。
- **%USED:** プールごとに使用されるストレージの概念パーセンテージ。
- **MAX AVAIL:** このプールに書き込むことができる想定データ量の推定。これは、最初の OSD がフルになる前に使用できるデータの量です。CRUSH マップからディスクにまたがるデータの予測分布を考慮し、フルにする最初の OSD をターゲットとして使用します。
  上記の例では、レプリケーション係数 (デフォルトでは 3) を考慮しない **MAX AVAIL** は 153.85 MB です。

  MAX AVAIL の値を計算するには、ceph df MAX AVAIL is incorrect for simple replicated pool というタイトルの Red Hat ナレッジベースの記事を参照してください。
- **QUOTA OBJECTS:** クォータオブジェクトの数。
- **QUOTA BYTES:** クォータオブジェクトのバイト数。
- **USED COMPR:** 圧縮データに割り当てられる領域の量これには、圧縮データ、割り当て、レプリケーション、およびイレイジャーコーディングのオーバーヘッドが含まれます。
- **UNDER COMPR:** 圧縮に渡されるデータの量で、圧縮された形式で保存することが十分に有益です。

注記

POOLS セクションの数字は概念的です。レプリカ、スナップショット、またはクローンの数は含まれていません。その結果、**USED** と **%USED** の量の合計は、出力の **GLOBAL** セクションの **RAW USED** と **%RAW USED** の量に加算されません。



注記

MAX AVAIL の値は、使用されるレプリケーションまたはイレイジャージャーコード、ストレージをデバイスにマッピングする CRUSH ルール、それらのデバイスの使用率、および設定された **mon_osd_full_ratio** の複雑な関数です。

関連情報

- 詳細は、Ceph のデータ使用量の計算方法 を参照してください。

- 詳細は、OSD の使用状況の統計について を参照してください。

3.1.5. OSD の使用状況の統計について

ceph osd df コマンドを使用して、OSD 使用率の統計を表示します。

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 の名前。
- **CLASS:** OSD が使用するデバイスのタイプ。
- **WEIGHT:** CRUSH マップの OSD の重み。
- **REWEIGHT:** デフォルトの再重み値です。
- **SIZE:** OSD の全体的なストレージ容量
- **USE:** OSD の容量
- **DATA:** ユーザーデータが使用する OSD 容量
- **OMAP:** オブジェクトマップ (**omap**) データを保存するために使用されている **bluefs** ストレージの推定値 (**rocksdb** に保存されたキー/値のペア)。
- **META:** 内部メタデータに割り当てられた **bluefs** の領域、または **bluestore_bluefs_min** パラメーターで設定された値のうちいずれか大きい方の値で、**bluefs** に割り当てられた領域の合計から推定 **omap** データサイズを差し引いた値として計算されます。
- **AVAIL:** OSD で利用可能な空き容量
- **%USE:** OSD で使用されるストレージの表記率
- **VAR:** 平均の使用率を上回るまたは下回る変動。
- **PGS:** OSD 内の配置グループ数
- **MIN/MAX VAR:** すべての OSD における変更の最小値および最大値。

関連情報

- 詳細は、**Ceph のデータ使用量の計算方法** を参照してください。
- 詳細は、**OSD の使用状況の統計について** を参照してください。
- 詳細は、**Red Hat Ceph Storage ストレージ戦略ガイド**の **CRUSH の重み** を参照してください。

3.1.6. ストレージクラスターのステータスの確認

コマンドラインインターフェイスから Red Hat Ceph Storage クラスターのステータスを確認できます。**status** サブコマンドまたは **-s** 引数は、ストレージクラスターの現在のステータスを表示します。

前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。
- ノードへのルートレベルのアクセス。

手順

1. Cephadm シェルにログインします。

   例

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

2. ストレージクラスターのステータスを確認するには、以下を実行します。

   例

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

   または、以下を実行します。

   例

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

3. インタラクティブモードで、**ceph** と入力し、**Enter** を押します。

   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. Ceph Monitor ステータスの確認

ストレージクラスターに複数の Ceph Monitor がある場合 (実稼働環境用の Red Hat Ceph Storage クラスターに必要)、ストレージクラスターの起動後に Ceph Monitor クォーラム (定足数) のステータスを確認し、データの読み取りまたは書き込みを実施する前に、Ceph Monitor クォーラムのステータスを確認します。

Ceph Monitor を実行している場合はクォーラムが存在する必要があります。

Ceph Monitor ステータスを定期的にチェックし、実行していることを確認します。Ceph Monitor に問題があり、ストレージクラスターの状態に関する合意ができない場合は、その障害により Ceph クライアントがデータを読み書きできなくなる可能性があります。

前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。
- ノードへのルートレベルのアクセス。

手順

1. Cephadm シェルにログインします。

   例

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

2. Ceph Monitor マップを表示するには、以下を実行します。

   例

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

   または、以下を実行します。

   例

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

3. ストレージクラスターのクォーラムステータスを確認するには、以下を実行します。

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

Ceph はクォーラムステータスを返します。

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. Ceph 管理ソケットの使用

管理ソケットを使用して、UNIX ソケットファイルを使用して、指定したデーモンと直接対話します。たとえば、ソケットを使用すると以下を行うことができます。

- ランタイム時に Ceph 設定をリスト表示します。
- Monitor に依存せずに直接ランタイムに設定値を設定します。これは、モニターが **ダウン** している場合に便利です。
- ダンプの履歴操作
- 操作優先度キューの状態をダンプします。
- 再起動しないダンプ操作
- パフォーマンスカウンターのダンプ

さらに、Ceph Monitor または OSD に関連する問題のトラブルシューティングを行う場合は、ソケットの使用に役立ちます。

デーモンが実行されていない場合でも、管理ソケットの使用を試みる際に以下のエラーが返されます。

```
Error 111: Connection Refused
```

重要

管理ソケットは、デーモンの実行中にのみ利用できます。デーモンを正常にシャットダウンすると、管理ソケットが削除されます。ただし、デーモンが突然終了すると、管理ソケットが永続化される可能性があります。

前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。
- ノードへのルートレベルのアクセス。

手順

1. Cephadm シェルにログインします。

   例

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

2. ソケットを使用するには、以下を実行します。

   構文

   ```
   ceph daemon MONITOR_ID COMMAND
   ```

   以下を置き換えます。

   - デーモンの **MONITOR_ID**
   - **COMMAND** を、実行するコマンドに置き換えます。指定のデーモンで利用可能なコマンドをリスト表示するには、**help** を使用します。
     Ceph Monitor のステータスを表示するには、以下を実行します。

     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"
}
```

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. または、ソケットファイルを使用して Ceph デーモンを指定します。

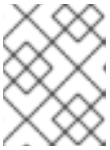
構文

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

4. 特定のホスト上の **osd.0** という名前の Ceph OSD のステータスを表示するには、次の手順を実行します。

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
}
```



注記

特定のデーモンで使用できるさまざまなオプションについては、**status** の代わりに **help** を使用できます。

5. Ceph プロセスのソケットファイルのリストを表示するには、以下のコマンドを実行します。

例

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

関連情報

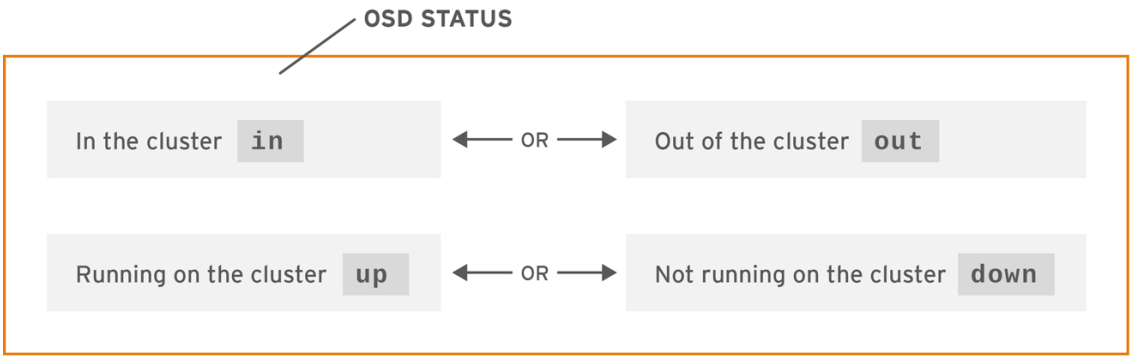
- 詳細は、Red Hat Ceph Storage トラブルシューティングガイドを参照してください。

3.1.9. Ceph OSD のステータスについて

Ceph OSD のステータスは、ストレージクラスター **in** またはストレージクラスター **out** のいずれかです。これは、**up** して稼働中か、**down** して実行されていないかのいずれかになります。Ceph OSD が **up** の場合は、データの読み取りおよび書き込みが可能なストレージクラスターにある (**in**) か、ストレージクラスターの外 (**out**) にあるかのいずれかになります。ストレージクラスター内 (**in**) にあり、最近ストレージクラスターの外 (**out**) に移動した場合、Ceph はプレースメントグループを他の Ceph OSD に移行し始めます。Ceph OSD がストレージクラスターの **out** にある場合、CRUSH は配置グループを Ceph OSD に割り当てません。Ceph OSD が **down** している場合は、それも **out** になっているはずです。

注記

Ceph OSD が **down** していて、**in** の場合は問題が発生しているため、ストレージクラスターは正常な状態になりません。



ceph health、**ceph -s**、**ceph -w** などのコマンドを実行すると、ストレージクラスターが常に **HEALTH OK** をエコーバックしないことが分かります。慌てないでください。Ceph OSD に関しては、いくつかの予想される状況では、ストレージクラスターが **HEALTH OK** をエコー **しない** ことが予想されます。

- ストレージクラスターをまだ開始しておらず、応答していません。
- ストレージクラスターを起動または再起動したばかりで、配置グループが作成されつつあり、Ceph OSD がピアリング中であるため、準備はできていません。
- Ceph OSD を追加または削除しました。
- ストレージクラスターマップを変更しました。

Ceph OSD の監視の重要な要素は、ストレージクラスターの起動時および稼働時にストレージクラスター内 (**in**) のすべての Ceph OSD が起動 (**up**) して稼働していることを確認することです。

すべての OSD が実行中かどうかを確認するには、以下を実行します。

例

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

または、以下を実行します。

例

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

結果により、マップのエポック (**eNNNN**)、OSD の総数 (**x**)、いくつの **y** が **up** で、いくつの **z** が **in** であるかが分かります。

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

ストレージクラスターにある (**in**) Ceph OSD の数が、稼働中 (**up**) の Ceph OSD 数を超える場合。以下のコマンドを実行して、実行していない **ceph-osd** デーモンを特定します。

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
```

ヒント

適切に設計された CRUSH 階層で検索する機能は、物理ロケーションをより迅速に特定してストレージクラスターをトラブルシューティングするのに役立ちます。

Ceph OSD がダウンしている (**down**) 場合は、ノードに接続して開始します。Red Hat Storage Console を使用して Ceph OSD デーモンを再起動するか、コマンドラインを使用できます。

構文

```
systemctl start CEPH_OSD_SERVICE_ID
```

例

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

関連情報

- 詳細は、Red Hat Ceph Storage ダッシュボードガイド を参照してください。

3.2. CEPH STORAGE クラスターの低レベルの監視

ストレージ管理者は、低レベルの視点から Red Hat Ceph Storage クラスターの正常性をモニターできます。通常、低レベルのモニタリングでは、Ceph OSD が適切にピアリングされるようにする必要があります。ピアの障害が発生すると、配置グループは動作が低下した状態で動作します。このパフォーマンスの低下状態は、ハードウェア障害、Ceph デーモンのハングまたはクラッシュした Ceph デーモン、ネットワークレイテンシー、完全なサイト停止など多くの異なる状態によって生じる可能性があります。

3.2.1. 配置グループセットの監視

CRUSH が配置グループを Ceph OSD に割り当てると、プールのレプリカ数を確認し、配置グループの各レプリカが別の Ceph OSD に割り当てられるように配置グループを Ceph OSD に割り当てます。たとえば、プールに配置グループの 3 つのレプリカが必要な場合、CRUSH はそれらをそれぞれ **osd.1**、**osd.2**、および **osd.3** に割り当てることができます。CRUSH は実際には、CRUSH マップで設定した障害ドメインを考慮した擬似ランダムな配置を探すため、大規模なクラスター内で最も近い Ceph OSD に割り当てられた配置グループを目にすることはほとんどありません。特定の配置グループのレプリカを **動作セット** として組み込む必要がある Ceph OSD のセットを参照します。場合によって

は、Acting Set の OSD が **down** になった場合や、配置グループ内のオブジェクトのリクエストに対応できない場合があります。このような状況が発生しても、慌てないでください。一般的な例を示します。

- OSD を追加または削除しています。次に、CRUSH は配置グループを他の Ceph OSD に再度割り当てます。これにより、動作セットの設定を変更し、バックフィルプロセスでデータの移行を生成します。
- Ceph OSD が **down** になり、再起動されてリカバリー中 (**recovering**) となっています。
- 動作セットの Ceph OSD は **down** となっているが、要求に対応できず、別の Ceph OSD がそのロールを一時的に想定しています。

Ceph は **Up Set** を使用してクライアント要求を処理します。これは、実際に要求を処理する Ceph OSD のセットです。ほとんどの場合、up set と Acting Set はほぼ同一です。そうでない場合には、Ceph がデータを移行しているか、Ceph OSD が復旧するか、問題がある場合に、通常 Ceph がこのようなシナリオで stuck stale メッセージと共に **HEALTH WARN** 状態を出すことを示しています。

前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。
- ノードへのルートレベルのアクセス。

手順

1. Cephadm シェルにログインします。

   例

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

2. 配置グループのリストを取得するには、次のコマンドを実行します。

   例

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

3. 特定の配置グループの **Acting Set**{b> <b}または **Up Set** にある OSD を表示します。

   構文

   ```
   ceph pg map PG_NUM
   ```

   例

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

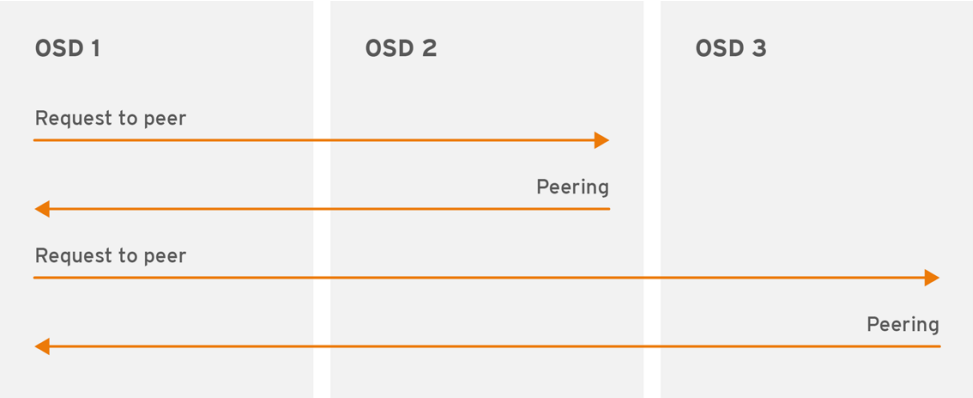
注記

Up Set と Acting Set が一致しない場合は、ストレージクラスター自体をリバランスするか、ストレージクラスターで潜在的な問題があることを示している可能性があります。

3.2.2. Ceph OSD のピアリング

配置グループにデータを書き込む前に、そのデータを **active** 状態にし、**clean** な状態で **なければなりません**。Ceph が配置グループの現在の状態を決定するためには、配置グループの第一 OSD (動作セットの最初の OSD) が、第二および第三の OSD とピアリングを行い、配置グループの現在の状態についての合意を確立します。PG のレプリカが 3 つあるプールを想定します。

図3.1 ピアリング



3.2.3. 配置グループの状態

ceph health、**ceph -s**、**ceph -w** などのコマンドを実行すると、クラスターが常に **HEALTH OK** をエコーバックしないことが分かります。OSD が実行中であるかを確認したら、配置グループのステータスも確認する必要があります。数多くの配置グループのピア関連状況で、クラスターが **HEALTH OK** を **しない** ことが予想されます。

- プールを作成したばかりで、配置グループはまだピアリングしていません。
- 配置グループは復旧しています。
- クラスターに OSD を追加したり、クラスターから OSD を削除したりしたところです。
- CRUSH マップを変更し、配置グループが移行中である必要があります。
- 配置グループの異なるレプリカに一貫性のないデータがあります。
- Ceph は配置グループのレプリカをスクラビングします。
- Ceph には、バックフィルの操作を完了するのに十分なストレージ容量がありません。

前述の状況のいずれかにより Ceph が **HEALTH WARN** をエコーしても慌てる必要はありません。多くの場合、クラスターは独自にリカバリーします。場合によっては、アクションを実行する必要がある場合があります。配置グループを監視する上で重要なことは、クラスターの起動時にすべての配置グループが **active** で、できれば **clean** な状態であることを確認することです。

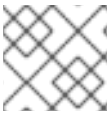
すべての配置グループのステータスを表示するには、以下を実行します。

例

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

その結果、配置グループマップバージョン (**vNNNNNN**)、配置グループの合計 (**x**)、および配置グループの数 (**y**) が、**active+clean** などの特定の状態にあることを示します。

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



注記

Ceph では、配置グループについて複数の状態を報告するのが一般的です。

スナップショットトリミングの PG の状態

スナップショットが存在する場合には、追加の PG ステータスが 2 つ報告されます。

- **snaptrim**: PG は現在トリミング中です。
- **snaptrim_wait**: PG はトリム処理を待機中です。

出力例:

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

Ceph は、配置グループの状態に加えて、使用データ量 (**aa**)、ストレージ容量残量 (**bb**)、配置グループの総ストレージ容量をエコーバックします。いくつかのケースでは、これらの数字が重要になります。

- **near full ratio** または **full ratio** に達しています。
- CRUSH 設定のエラーにより、データがクラスター全体に分散されません。

配置グループ ID

配置グループ ID は、プール名ではなくプール番号で設定され、ピリオド (.) と配置グループ ID が続きます (16 進数)。**ceph osd lspools** の出力で、プール番号およびその名前を表示することができます。デフォルトのプール名 **data**、**metadata**、**rbd** はそれぞれプール番号 **0**、**1**、**2** に対応しています。完全修飾配置グループ ID の形式は以下のとおりです。

構文

```
POOL_NUM.PG_ID
```

出力例:

```
0.1f
```

- 配置グループのリストを取得するには、次のコマンドを実行します。

  例

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

- JSON 形式で出力をフォーマットし、ファイルに保存するには、以下を実行します。

  構文

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

例

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

- 以下のように、特定の配置グループにクエリーを行います。

構文

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

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",
```

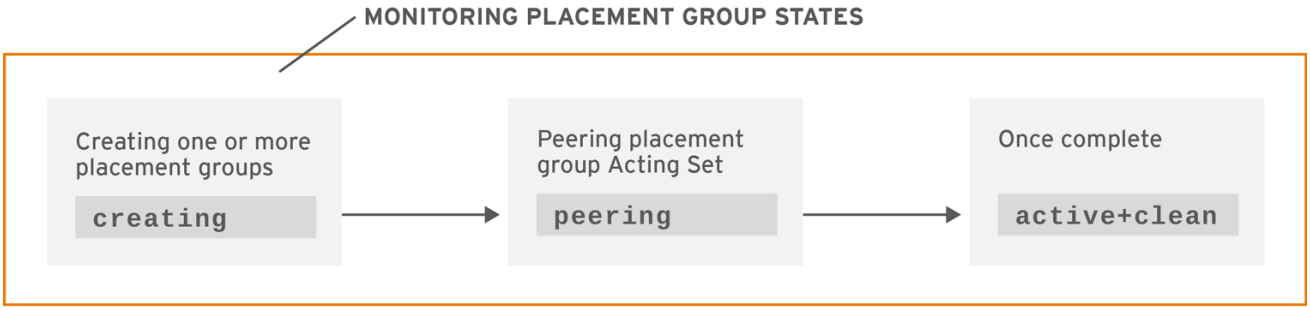
```
            "last_deep_scrub": "0'0",
            "last_deep_scrub_stamp": "2021-06-17T01:32:03.763988+0000",
            "last_clean_scrub_stamp": "2021-06-17T01:32:03.763988+0000",
            "prior_readable_until_ub": 0
        },
        "stats": {
            "version": "0'0",
            "reported_seq": "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",
....
```

関連情報

- スナップショットトリミングの設定に関する詳細は、**Red Hat Ceph Storage 設定ガイド** の OSD Object ストレージデーモン設定オプション の **Object Storage Daemon (OSD) の設定オプション** を参照してください。

3.2.4. 配置グループの状態の作成

プールを作成すると、指定した数の配置グループが作成されます。Ceph は、1 つ以上の配置グループの **作成** 時に作成をエコーします。これが作成されると、配置グループのアクティングセットの一部である OSD がピアリングを行います。ピアリングが完了すると、配置グループのステータスは **active+clean** になり、Ceph クライアントが配置グループへの書き込みを開始できるようになります。



CEPH_459704_1017

3.2.5. 配置グループのピア状態

Ceph が配置グループをピアリングする場合、Ceph は配置グループのレプリカを保存する OSD を配置グループ内のオブジェクトおよびメタデータの **状態について合意** に持ち込みます。Ceph がピアリングを完了すると、配置グループを格納する OSD が配置グループの現在の状態について合意することを意味します。ただし、ピアリングプロセスを完了しても、各レプリカに最新のコンテンツがある **わけではありません**。

権威の履歴

Ceph は、動作セットのすべての OSD が書き込み操作を持続させるまで、クライアントへの書き込み操作を **承認しません**。これにより、有効なセットの少なくとも 1 つメンバーが、最後に成功したピア操作以降の確認済みの書き込み操作がすべて記録されるようになります。

それぞれの確認応答書き込み操作の正確なレコードにより、Ceph は配置グループの新しい権威履歴を構築して公開することができます。完全かつ完全に命令された一連の操作が実行されれば、OSD の配置グループのコピーを最新の状態にすることができます。

3.2.6. 配置グループのアクティブな状態

Ceph がピア処理を完了すると、配置グループが **active** になる可能性があります。**active** 状態とは、配置グループのデータがプライマリー配置グループで一般的に利用可能で、読み取り操作および書き込み操作用のレプリカになります。

3.2.7. 配置グループの clean の状態

配置グループが **クリーン** な状態にある場合、プライマリー OSD とレプリカ OSD は正常にピアリングを行い、配置グループ用の迷子のレプリカが存在しないことを意味します。Ceph は、配置グループ内のすべてのオブジェクトを正しい回数で複製します。

3.2.8. 配置グループの状態が低下した状態

クライアントがプライマリー OSD にオブジェクトを書き込む際に、プライマリー OSD はレプリカ OSD にレプリカを書き込むロールを担います。プライマリー OSD がオブジェクトをストレージに書き込んだ後に、配置グループは、Ceph がレプリカオブジェクトを正しく作成したレプリカ OSD からプライマリー OSD が確認応答を受け取るまで、動作が **低下** した状態になります。

配置グループが **active+degraded** になる理由は、OSD がまだすべてのオブジェクトを保持していない場合でも **active** である可能性があることです。OSD が **down** する場合、Ceph は OSD に割り当てられた各配置グループを **degraded** としてマークします。Ceph OSD がオンラインに戻る際に、Ceph OSD は再度ピアリングする必要があります。ただし、クライアントは、**active** であれば、**degraded** である配置グループに新しいオブジェクトを記述できます。

OSD が **down** していてパフォーマンスの低下 (**degraded**) が続く場合には、Ceph は **down** 状態である OSD をクラスターの外 (**out**) としてマークし、**down** 状態である OSD から別の OSD にデータを再マッピングする場合があります。**down** とマークされた時間と **out** とマークされた時間の間の時間は **mon_osd_down_out_interval** によって制御され、デフォルトでは **600** に設定されています。

また、配置グループは、Ceph が配置グループにあるべきだと考えるオブジェクトを 1 つ以上見つけることができないため、**低下** してしまうこともあります。未検出オブジェクトへの読み取りまたは書き込みはできませんが、動作が低下した (**degraded**) 配置グループ内の他のすべてのオブジェクトにアクセスできます。

たとえば、3 方向のレプリカプールに 9 つの OSD があるとします。OSD の数の 9 がダウンすると、9 の OSD に割り当てられた PG は動作が低下します。OSD 9 がリカバリーされない場合は、ストレージクラスターから送信され、ストレージクラスターがリバランスします。このシナリオでは、PG のパフォーマンスが低下してから、アクティブな状態に戻ります。

3.2.9. 配置グループの状態のリカバリー

Ceph は、ハードウェアやソフトウェアの問題が継続している規模でのフォールトトレランスを目的として設計されています。OSD がダウンする (**down**) と、そのコンテンツは配置グループ内の他のレプリカの現在の状態のままになる可能性があります。OSD が **up** 状態に戻ったら、配置グループの内容を更新して、現在の状態を反映させる必要があります。その間、OSD は **リカバリー** の状態を反映する場合があります。

ハードウェアの故障は、複数の Ceph OSD のカスケード障害を引き起こす可能性があるため、回復は常に些細なことではありません。たとえば、ラックやキャビネット用のネットワークスイッチが故障して、多数のホストマシンの OSD がストレージクラスターの現在の状態から遅れてしまうことがありま

す。各 OSD は、障害が解決されたら回復しなければなりません。

Ceph は、新しいサービス要求とデータオブジェクトの回復と配置グループを現在の状態に復元するニーズの間でリソース競合のバランスを取るためのいくつかの設定を提供しています。**osd recovery delay start** 設定により、回復プロセスを開始する前に OSD を再起動し、ピアリングを再度行い、さらにはいくつかの再生要求を処理できます。**osd recovery threads** 設定により、デフォルトで 1 つのスレッドでリカバリープロセスのスレッド数が制限されます。**osd recovery thread timeout** は、複数の Ceph OSD が驚きの速さで失敗、再起動、再ピアする可能性があるため、スレッドタイムアウトを設定します。**osd recovery max active** 設定では、Ceph OSD が送信に失敗するのを防ぐために Ceph OSD が同時に実行するリカバリー要求の数を制限します。**osd recovery の max chunk** 設定により、復元されたデータチャンクのサイズが制限され、ネットワークの輻輳を防ぐことができます。

3.2.10. バックフィルの状態

新規 Ceph OSD がストレージクラスターに参加する際に、CRUSH はクラスター内の OSD から新たに追加された Ceph OSD に配置グループを再割り当てします。新規 OSD が再割り当てされた配置グループをすぐに許可するように強制すると、新規 Ceph OSD に過剰な負荷が生じる可能性があります。OSD を配置グループでバックフィルすると、このプロセスはバックグラウンドで開始できます。バックフィルが完了すると、新しい OSD の準備が整い次第、要求への対応を開始します。

バックフィル操作中に、次のいずれかの状態が表示される場合があります。

- **backfill_wait** は、バックフィル操作が保留中であるが、まだ進行中でないことを示します
- **backfill** は、バックフィル操作が進行中であることを示します
- **backfill_too_full** は、バックフィル操作が要求されたが、ストレージ容量が不十分なために完了できなかったことを示します。

配置グループをバックフィルできない場合は、**incomplete** とみなされることがあります。

Ceph は、Ceph OSD、とくに新しい Ceph OSD への配置グループの再割り当てに関連する負荷の急増を管理するための数多くの設定を提供しています。デフォルトでは、**osd_max_backfills** は、Ceph OSD から 10 への同時バックフィルの最大数を設定します。**osd backfill full ratio** により、Ceph OSD は、OSD が完全な比率 (デフォルトでは 85%) に近づけている場合にバックフィル要求を拒否することができます。OSD がバックフィル要求を拒否する場合は、**osd backfill retry interval** により、OSD はデフォルトで 10 秒後に要求を再試行できます。また、OSD は、スキャン間隔 (デフォルトで 64 および 512) を管理するために、**osd backfill scan min** および **osd backfill scan max** を設定することもできます。

ワークロードによっては、通常のリカバリーを完全に回避し、代わりにバックフィルを使用することが推奨されます。バックフィルはバックグラウンドで実行されるため、I/O は OSD のオブジェクトで続行できます。**osd_min_pg_log_entries** オプションを **1** に設定し、**osd_max_pg_log_entries** オプションを **2** に設定することで、リカバリーではなくバックフィルを強制できます。この状況がご使用のワークロードに適切な場合についての詳細は、Red Hat サポートアカウントチームにお問い合わせください。

3.2.11. 配置グループの再マッピングの状態

配置グループにサービスを提供する動作セットが変更すると、古い動作セットから新しい動作セットにデータを移行します。新規プライマリー OSD がリクエストを処理するには、多少時間がかかる場合があります。したがって、配置グループの移行が完了するまで、古いプライマリーに要求への対応を継続するように依頼する場合があります。データの移行が完了すると、マッピングは新しい動作セットのプライマリー OSD を使用します。

3.2.12. 配置グループの stale 状態

Ceph はハートビートを使用してホストとデーモンが実行されていることを確認しますが、**ceph-osd** デーモンも **スタック** 状態になり、統計をタイムリーに報告しない場合があります。たとえば、一時的なネットワーク障害などが挙げられます。デフォルトでは、OSD デーモンは、配置グループ、アップスルー、ブート、失敗の統計情報を半秒 (**0.5**) ごとに報告しますが、これはハートビートのしきい値よりも頻度が高くなります。配置グループの動作セットの **プライマリー OSD** がモニターへの報告に失敗した場合や、他の OSD がプライマリー OSD の **down** を報告した場合、モニターは配置グループに **stale** マークを付けます。

ストレージクラスターを起動すると、ピアリング処理が完了するまで **stale** 状態になるのが一般的です。ストレージクラスターがしばらく稼働している間に、配置グループが **stale** 状態になっているのが確認された場合は、その配置グループのプライマリー OSD が **down** になっているか、モニターに配置グループの統計情報を報告していないことを示しています。

3.2.13. 配置グループの不配置の状態

PG が OSD に一時的にマップされる一時的なバックフィルシナリオがあります。**一時的** な状況がなくなった場合には、PG は一時的な場所に留まり、適切な場所にない可能性があります。いずれの場合も、それらは **誤って配置** されます。それは、実際には正しい数の追加コピーが存在しているのに、1 つ以上のコピーが間違った場所にあるためです。

たとえば、3 つの OSD が 0、1、2 であり、すべての PG はこれらの 3 つのうちのいくつかの配列にマップされます。別の OSD (OSD 3) を追加する場合、一部の PG は、他のものではなく OSD 3 にマッピングされるようになりました。しかし、OSD 3 がバックフィルされるまで、PG には一時的なマッピングがあり、古いマッピングからの I/O を提供し続けることができます。その間、PG には一時的な待っピンがありますが、コピーが 3 つあるため **degraded** はしていないため、間違った場所に置かれます (**misplaced**)。

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] は一時的なマッピングです。そのため、**up** セットは **acting** なセットとは等しくならず、[0,1,2] がまだ 3 つのコピーであるため、PG は誤って配置されます (**misplaced**) が、パフォーマンスは低下 (**degraded**) しません。

例

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

OSD 3 はバックフィルされ、一時的なマッピングは削除され、パフォーマンスは低下せず、誤って配置されなくなりました。

3.2.14. 配置グループの不完全な状態

PG は、不完全なコンテンツがあり、ピアリングが失敗したとき、すなわち、リカバリーを実行するためのな現在の完全な OSD が十分にないときに、**incomplete** 状態になる。

例えば、OSD 1、2、3 が動作中の OSD セットで、それが OSD 1、4、3 に切り替わったとすると、**osd.1** が 4 をバックフィルする間、OSD 1、2、3 の一時的な動作中の OSD セットを要求することになリマス。この間、OSD 1、2、および 3 すべてがダウンすると、**osd.4** は、すべてのデータが完全に

バックフィルされていない可能性がある唯一のものとして残されています。このとき、PG は **incomplete** となり、リカバリーを実行するのに十分な現在の完全な OSD がないことを示す不完全な状態になります。

別の方法として、**osd.4** が関与しておらず、OSD の 1、2、3 がダウンしたときに動作セットが単に OSD 1、2、3 になっている場合、PG はおそらく動作セットが変更されてからその PG で何も聞いていないことを示す **stale** になります。新規 OSD に通知する OSD がない理由。

3.2.15. スタックした配置グループの特定

配置グループは、**active+clean** 状態ではないという理由だけで必ずしも問題になるとは限りません。一般的に、Ceph の自己修復機能は、配置グループが停止する場合に機能しない場合があります。スタック状態には、以下が含まれます。

- **Unclean**: 配置グループには、必要な回数複製しないオブジェクトが含まれます。これらは回復中である必要があります。
- **Inactive**: 配置グループは、最新のデータを持つ OSD が **up** に戻るのを待っているため、読み取りや書き込みを処理できません。
- **Stale**: 配置グループは不明な状態です。配置グループは、これらをホストする OSD がしばらくモニタークラスターに報告されず、**mon osd report timeout** 設定で設定できるためです。

前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。
- ノードへのルートレベルのアクセス。

手順

1. スタックした配置グループを特定するには、以下のコマンドを実行します。

   構文

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

   例

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

3.2.16. オブジェクトの場所の検索

Ceph クライアントは最新のクラスターマップを取得し、CRUSH アルゴリズムはオブジェクトを配置グループにマッピングする方法を計算してから、配置グループを OSD に動的に割り当てる方法を計算します。

前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。
- ノードへのルートレベルのアクセス。

手順

1. オブジェクトの場所を見つけるには、オブジェクト名とプール名のみが必要です。

   構文

   ```
   ceph osd map POOL_NAME OBJECT_NAME
   ```

   例

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

第4章 CEPH STORAGE のストレッチクラスター

ストレージ管理者は、2 サイトクラスターでストレッチモードを開始することにより、ストレッチクラスターを設定できます。

Red Hat Ceph Storage は、CRUSH マップ全体にランダムに分散された障害に対して同等に信頼できるネットワークとクラスターにより、Ceph OSD の損失に耐えることができます。多くの OSD がシャットダウンされても、残りの OSD とモニターは引き続き動作します。

ただし、これは、Ceph クラスターの大部分が 1 つのネットワークコンポーネントしか使用できない一部のストレッチクラスター設定では最適なソリューションではない可能性があります。この例は、複数のデータセンターに配置された 1 つのクラスターについて、ユーザーがデータセンター全体の損失に耐えたいと考えている場合です。

標準設定は 2 つのデータセンターです。その他の設定は、クラウドまたは可用性ゾーンにあります。各サイトはデータのコピーを 2 つ保持するため、レプリケーションサイズは 4 です。3 番目のサイトには、タイブレーカーモニターが必要です。これは、メインサイトと比較して、仮想マシンまたは高レイテンシーである可能性があります。このモニターは、ネットワーク接続に障害が発生し、両方のデータセンターがアクティブなままである場合、データを復元するサイトの 1 つを選択します。

注記

標準の Ceph 設定は、ネットワークまたはデータセンターの多くの障害に耐え、データの一貫性を損なうことはありません。障害後に十分な数の Ceph サーバーを復元すると、回復します。Ceph は、データセンターを失った場合も、可用性を維持しますが、モニターの定足数を形成し、プールの **min_size** を満たすために十分なコピー、またはサイズを満たすために再度複製する CRUSH ルールで、すべてのデータを利用可能にすることができます。

注記

ストレッチクラスターの電源を切るために追加の手順はありません。詳細は、**Red Hat Ceph Storage クラスターの電源切断と再起動** を参照してください。

ストレッチクラスターの障害

Red Hat Ceph Storage は、データの整合性と一貫性について決して妥協しません。ネットワーク障害またはノードの損失があり、サービスを復元できる場合、Ceph は単独で通常の機能に戻ります。

ただし、Ceph の一貫性とサイジングの制約を満たすために十分な数のサーバーを使用できる場合も、データの可用性が失われる状況や、予想外に制約を満たさない状況があります。

最初の重要なタイプの障害は、一貫性のないネットワークによって引き起こされます。ネットワークが分割されている場合は、プライマリー OSD がデータをレプリケーションできないにもかかわらず、Ceph が OSD を **down** とマークして、代理配置グループ (PG) セットから削除できない場合があります。これが発生すると、Ceph が耐久性の保証を満たすことができないため、I/O は許可されません。

失敗の 2 番目の重要なカテゴリーは、データ入力間でデータがレプリケーションされているように見えるが、制約がこれを保証するには不十分な場合です。たとえば、データセンター A と B があり、CRUSH ルールは 3 つのコピーを対象とし、**min_size** が **2** の各データセンターにコピーを配置するとします。PG は、サイト A に 2 つのコピーがあり、サイト B にコピーがない状態でアクティブになる場合があります。つまり、サイト A を失うと、データが失われ、Ceph は、そのデータを操作できなくなります。この状況は、標準の CRUSH ルールでは、回避が困難です。

4.1. ストレージクラスターのストレッチモード

ストレッチクラスターを設定するには、ストレッチモードに入る必要があります。ストレッチモードが有効になっている場合、Ceph OSD は、PG がデータセンター間でピア接続している場合、または指定した他の CRUSH バケットタイプが両方ともアクティブであると仮定して、PG のみをアクティブと見なします。プールのサイズはデフォルトの 3 から 4 に増加し、各サイトに 2 つのコピーがあります。

ストレッチモードでは、Ceph OSD は同じデータセンター内のモニターのみに接続できます。新しいモニターは、場所を指定しないと、クラスターに参加できません。

データセンターのすべての OSD とモニターが一度にアクセスできなくなった場合、存続しているデータセンターは **degraded** ストレッチモードに入ります。これにより、警告が発行され、**min_size** が **1** に減少し、クラスターが残りのサイトからのデータで **active** 状態に到達できるようになります。

注記

プールサイズが変更されないため、**degraded** 状態でも、プールが小さすぎるという警告がトリガーされます。ただし、特別なストレッチモードフラグにより、OSD が残りのデータセンターに余分なコピーを作成することが防止されるため、2 つのコピーが保持されます。

欠落しているデータセンターが再びアクセス可能になると、クラスターは **recovery** ストレッチモードに入ります。これにより、警告が変更され、ピアリングが許可されますが、必要なものは、データセンターからの OSD のみであり、ずっと稼働していました。

すべての PG が既知の状態にあり、劣化でも不完全でもない場合、クラスターは通常のストレッチモードに戻り、警告を終了し、**min_size** を開始値 **2** に戻します。クラスターは、常に稼働していたサイトだけでなく、両方のサイトをピアリングする必要があるため、必要に応じて、他のサイトにフェイルオーバーできます。

ストレッチモードの制限

- 一度ストレッチモードに入ると、解除できません。
- ストレッチモードのクラスターでイレイジャーコーディングされたプールを使用することはできません。イレイジャーコーディングされたプールでストレッチモードに入ることも、ストレッチモードがアクティブな場合にイレイジャーコーディングされたプールを作成することもできません。
- 2 サイト以下のストレッチモードがサポートされています。
- 2 つのサイトの重みは同じである必要があります。そうでない場合は、次のエラーが表示されます。

  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!
  ```

  両方のサイトで同じ重みを実現するには、2 つのサイトにデプロイされた Ceph OSD のサイズが同じである必要があります。つまり、最初のサイトのストレージ容量は 2 番目のサイトのストレージ容量と同じです。

- 強制ではありませんが、各サイトで 2 つの Ceph モニターを実行し、タイブレーカーを 1 つ、合計 5 つ実行する必要があります。これは、ストレッチモードの場合、OSD が自分のサイトのモニターにしか接続できないためです。
- 独自の CRUSH ルールを作成する必要があります。これにより、各サイトに 2 つのコピーが提供され、両方のサイトで合計 4 つになります。
- デフォルト以外のサイズまたは **min_size** の既存のプールがある場合は、ストレッチモードを有効にすることはできません。
- クラスターは劣化時に **min_size 1** で実行されるため、オールフラッシュ OSD ではストレッチモードのみを使用する必要があります。これにより、接続が復元された後の回復に必要な時間が最小限に抑えられ、データ損失の可能性が最小限に抑えられます。

関連情報

- トラブルシューティングの手順は、**ストレッチモードでのクラスターのトラブルシューティング** を参照してください。

4.1.1. CRUSH の場所をデーモンに設定する

ストレッチモードに入る前に、CRUSH の場所を Red Hat Ceph Storage クラスターのデーモンに設定して、クラスターを準備する必要があります。これには 2 つの方法があります。

- サービス設定ファイルを介してクラスターをブートストラップします。このファイルでは、配置の一部として場所がホストに追加されます。
- クラスターがデプロイされた後、**ceph osd crush add-bucket** および **ceph osd crush move** コマンドを使用して、場所を手動で設定します。

方法 1: クラスターのブートストラップ

前提条件

- ノードへの root レベルのアクセス。

手順

1. 新しいストレージクラスターをブートストラップする場合は、ノードを Red Hat Ceph Storage クラスターに追加し、サービスを実行する場所に特定のラベルを設定するサービス設定 **.yaml** ファイルを作成できます。

   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
```

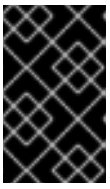
2. **--apply-spec** オプションを使用してストレージクラスターをブートストラップします。

構文

```
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
```

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

cephadm bootstrap コマンドでは、さまざまなコマンドオプションを使用できます。ただし、サービス設定ファイルを使用して、ホストの場所を設定するには、**--apply-spec** オプションを常に含めてください。

関連情報

- Ceph ブートストラップおよびさまざまな **cephadm bootstrap** コマンドオプションの詳細は、**新しいストレージクラスターのブートストラップ** を参照してください。

方法 2: デプロイメント後に場所を設定する

前提条件

- ノードへの root レベルのアクセス。

手順

1. 非タイブレーカーモニターの場所を設定する予定の 2 つのバケットを CRUSH マップに追加し、バケットタイプを **datacenter** として指定します。

   構文

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

   例

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

2. **root=default** の下にバケットを移動します。

   構文

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

   例

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

3. 必要な CRUSH 配置に従って、OSD ホストを移動します。

   構文

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

   例

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

4.1.2. ストレッチモードに入る

新しいストレッチモードは、2 つのサイトを処理するように、設計されています。2 サイトクラスターでは、コンポーネントの可用性が失われるリスクが低くなります。

前提条件

- ノードへの root レベルのアクセス。

- CRUSH の場所はホストに設定されます。

手順

1. CRUSH マップに合わせて、各モニターの位置を設定します。

   構文

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

   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. 各データセンターに 2 つのコピーを配置する CRUSH ルールを生成します。

   構文

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

   例

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

   a. 逆コンパイルされた CRUSH マップファイルを編集して、新しいルールを追加します。

      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) ルール **id** は一意である必要があります。この例では、**ID 0** のルールがもう 1 つしかないため、**ID 1** が使用されますが、既存のルールの数に応じて、別のルール ID を使用する必要がある場合があります。

2 3 この例では、**DC1** および **DC2** という名前の 2 つのデータセンターバケットがあります。

注記

このルールにより、クラスターはデータセンター **DC1** に対して読み取りアフィニティーを持ちます。したがって、すべての読み取りまたは書き込みは、**DC1** に配置された Ceph OSD を介して行われます。

これが望ましくなく、読み取りまたは書き込みがゾーン全体に均等に分散される場合、CRUSH ルールは次のようになります。

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
}
```

このルールでは、データセンターはランダムかつ自動的に選択されます。

firstn および **indep** オプションの詳細は、CRUSH ルール を参照してください。

3. CRUSH マップを挿入して、クラスターでルールを使用できるようにします。

構文

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

例

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

4. 接続モードでモニターを実行しない場合は、選択戦略を **connectivity** に設定します。

例

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

5. タイブレーカーモニターの場所をデータセンター間で分割するように設定して、ストレッチモードに入ります。

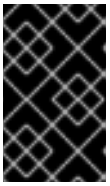
構文

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

例

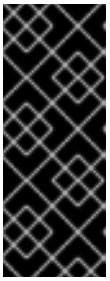
```
[ceph: root@host01 /]# ceph mon set_location host07 datacenter=DC3
[ceph: root@host01 /]# ceph mon enable_stretch_mode host07 stretch_rule datacenter
```

この例では、モニター **mon.host07** がタイブレーカーです。



重要

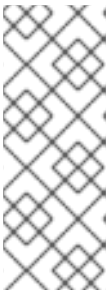
タイブレーカーモニターの場所は、以前に非タイブレーカーモニターを設定したデータセンターとは異なる必要があります。上記の例では、データセンター **DC3** です。



重要

ストレッチモードに入ろうとすると、次のエラーが発生するため、このデータセンターを CRUSH マップに追加しないでください。

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



注記

Ceph をデプロイするための独自のツールを作成している場合、**ceph mon set_location** コマンドを実行する代わりに、モニターの起動時に新しい **--set-crush-location** オプションを使用できます。このオプションは、**ceph-mon --set-crush-location 'datacenter=DC1'** など、1 つの **bucket=location** ペアのみを受け入れます。これは、**enable_stretch_mode** コマンドの実行時に指定したバケットタイプに一致する必要があります。

6. ストレッチモードが正常に有効になっていることを確認します。

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
```

Stretch_mode_enabled は **true** に設定する必要があります。また、ストレッチバケット、ストレッチモードバケットの数、およびストレッチモードが低下しているか回復しているかを確認することもできます。

7. モニターが適切な場所にあることを確認します。

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
```

また、どのモニターがタイブレーカーであるか、およびモニターの選択戦略も確認できます。

関連情報

- モニターの選択戦略の詳細は、**モニターの選択戦略の設定** を参照してください。

4.1.3. ストレッチモードでの OSD ホストの追加

ストレッチモードで Ceph OSD を追加できます。この手順は、ストレッチモードが有効になっていないクラスターに OSD ホストを追加する場合に類似しています。

前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。
- クラスターでストレッチモードが有効になっている。
- ノードへの root レベルのアクセス。

手順

1. OSD をデプロイするために利用可能なデバイスをリスト表示します。

   構文

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

   例

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

2. OSD を特定のホストまたは使用可能なすべてのデバイスにデプロイします。

   - 特定のホスト上の特定のデバイスから OSD を作成します。

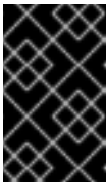
     構文

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

     例

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

   - 使用可能なデバイスと未使用のデバイスに OSD をデプロイします。

     重要

     このコマンドは、コロケーションされた WAL および DB デバイスを作成します。コロケーションされていないデバイスを作成する場合は、このコマンドを使用しないでください。

     例

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

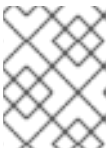
3. OSD ホストを CRUSH バケットの下に移動します。

   構文

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

   例

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

   注記

   両方のサイトに同じトポロジーノードを追加してください。ホストが 1 つのサイトのみに追加されると、問題が発生する可能性があります。

関連情報

- Ceph OSD の追加の詳細については、**OSD の追加** を参照してください。

第5章 CEPH の動作のオーバーライド

ストレージ管理者は、ランタイム時に Red Hat Ceph Storage クラスターのオーバーライドを使用して Ceph オプションを変更する方法を理解する必要があります。

5.1. CEPH のオーバーライドオプションの設定および設定解除

Ceph のデフォルト動作を上書きするために、Ceph オプションを設定および設定解除することができます。

前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。
- ノードへのルートレベルのアクセス。

手順

1. Ceph のデフォルトの動作を上書きするには、**ceph osd set** コマンドおよび上書きする動作を使用します。

   構文

   ```
   ceph osd set FLAG
   ```

   動作を設定したら、**ceph health** には、クラスターに設定したオーバーライドが反映されます。

   例

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

2. Ceph のデフォルトの動作を上書きするには、**ceph osd unset** コマンドおよび停止するオーバーライドを使用します。

   構文

   ```
   ceph osd unset FLAG
   ```

   例

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

フラグ	説明
noin	OSD が、クラスター **内** での扱いになるのを防ぎます。
noout	OSD が、クラスター **外** での扱いになるのを防ぎます。
noup	OSD が **up** で稼働している扱いになるのを防ぎます。

フラグ	説明
nodown	OSD が **down** 扱いされるのを防ぎます。
full	クラスターが **full_ratio** に達したように表示され、書き込み操作が阻止されます。
pause	Ceph は読み取り操作および書き込み操作の処理を停止しますが、ステータスの OSD のステータス **in**、**out**、**up**、または **down** は影響を受けません。
nobackfill	Ceph により、新しいバックフィルの操作が回避されます。
norebalance	Ceph により、新たなリバランス操作が回避されます。
norecover	Ceph により、新たなリカバリー操作が回避されます。
noscrub	Ceph は新規スクラビングの操作を回避します。
nodeep-scrub	Ceph は、新たな深層スクラブ作業を行いません。
notieragent	Ceph は、コールド/ダーティーオブジェクトをフラッシュしてエビクトすることを目的とするプロセスを無効にします。

5.2. CEPH のオーバーライドのユースケース

- **noin**: フラッピング OSD に対応するために、多くの場合 **noout** と一緒に使用されます。
- **noout**: **mon osd report timeout** を超え、OSD がモニターに報告されていない場合には、OSD は **out** とマークされます。誤って発生する場合は、問題のトラブルシューティング中に OSD が **out** とマークされないように **noout** を設定できます。
- **noup**: 一般的に、**nodown** で使用され、フラグッピング OSD に対応します。
- **nodown**: ネットワークの問題が Ceph の heartbeat プロセスが中断する可能性があり、OSD が **up** にある可能性がありますが、down をマークされる場合もあります。**nodown** を設定すると、問題のトラブルシューティング中に OSD が down をマークされないようにできます。
- **full**: クラスターが **full_ratio** に到達する場合は、事前にクラスターを **full** に設定し、容量を拡張することができます。

注記

クラスターを **full** に設定すると書き込み操作ができなくなります。

- **pause**: クライアントがデータの読み取りおよび書き込みを行わずに実行中の Ceph クラスターをトラブルシューティングする必要がある場合は、クライアントの操作を防ぐためにクラスターを **一時停止** するように設定できます。
- **nobackfill**: OSD またはノードを一時的に **down** する必要がある場合 (デーモンのアップグレードなど) は、OSD が **down** になっている間に Ceph がバックフィルしないように、**nobackfill** を設定できます。

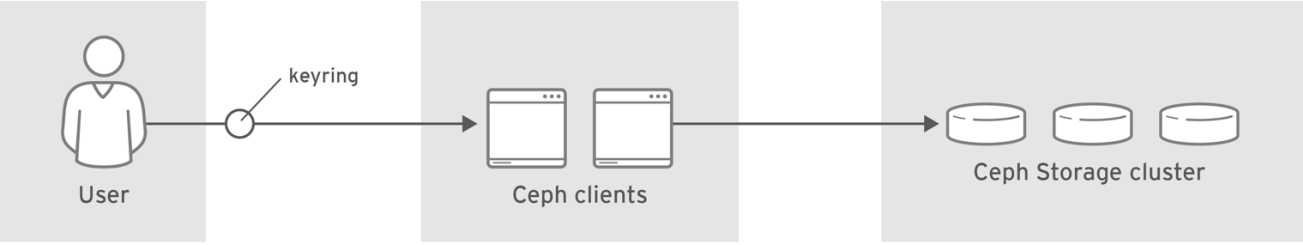
- **norecover**: OSD を置き換える必要がえあり、ディスクをホットスワップする間に PG を別の OSD に復元しないようする場合は、他の OSD のセットが他の OSD に新しいセットをコピーしないように、**norecover** も設定できます。

- **noscrub** および **nodeep-scrubb**: たとえば、高負荷、復旧、バックフィル、およびリバランス中のオーバーヘッドを減らすためにスクラビングを防ぐために、**noscrub** や **nodeep-scrub** を設定して、クラスターが OSD をスクラビングしないようにすることができます。

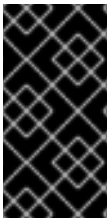
- **notieragent**: 階層エージェントプロセスで、バッキングストレージ層にコールドオブジェクトを検索しないようにするには、**notieragent** を設定する可能性があります。

第6章 CEPH ユーザー管理

ストレージ管理者は、Red Hat Ceph Storage クラスターのオブジェクトへの認証およびアクセス制御を提供することで、Ceph ユーザーのベースを管理できます。



CEPH_459704_1017



重要

クライアントが Cephadm の範囲内にある限り、Cephadm は Red Hat Ceph Storage クラスターのクライアントキーリングを管理します。トラブルシューティングを行わない限り、ユーザーは Cephadm によって管理されているキーリングを変更しないでください。

6.1. CEPH ユーザー管理の背景

認証と承認を有効にして Ceph を実行する場合は、ユーザー名を指定する必要があります。ユーザー名を指定しない場合、Ceph は **client.admin** 管理ユーザーをデフォルトのユーザー名として使用します。

ユーザー名およびシークレットの再入力を避けるために、**CEPH_ARGS** 環境変数を使用できます。

Ceph クライアントのタイプ (ブロックデバイス、オブジェクトストア、ファイルシステム、ネイティブ API、Ceph コマンドラインなど) に関係なく、Ceph はすべてのデータをオブジェクトとしてプールに保存します。データの読み取りおよび書き込みを行うには、Ceph ユーザーはプールにアクセスできる必要があります。また、管理用 Ceph ユーザーには、Ceph の管理コマンドを実行するパーミッションが必要です。

Ceph ユーザー管理の概念は以下のとおりです。

ストレージクラスターユーザー

Red Hat Ceph Storage クラスターのユーザーは、個別またはアプリケーションです。ユーザーを作成することで、誰がストレージクラスター、そのプール、およびそれらのプール内のデータにアクセスできるかを制御することができます。

Ceph の概念にはユーザーの **タイプ** があります。ユーザー管理の目的で、タイプは常に **client** になります。Ceph は、ユーザータイプとユーザー ID で設定されるピリオド (.) で区切られたユーザーを識別します。たとえば、**TYPE.ID**、**client.admin**、**client.user1** などです。ユーザーの入力は、Ceph Monitor および OSD も Cephx プロトコルを使用しますが、それらはクライアントではないために必要になります。ユーザータイプの分類することにより、クライアントユーザーと他のユーザーを区別でき、アクセス制御、ユーザーの監視および追跡可能性をさらに単純化します。

Ceph コマンドラインを使用すると、コマンドラインでの使用に応じて、タイプを使用せずにユーザーを指定できるため、Ceph のユーザータイプが混乱する場合があります。**--user** または **--id** を指定した場合は、タイプを省略できます。そのため、**client.user1** は **user1** として簡単に入力できます。**--name** または **-n** を指定する場合は、**client.user1** などのタイプおよび名前を指定する必要があります。Red Hat は、可能な限り、タイプと名前をベストプラクティスとして使用することを推奨します。

注記

Red Hat Ceph Storage クラスターユーザーは、Ceph Object Gateway ユーザーと同じではありません。オブジェクトゲートウェイは Red Hat Ceph Storage クラスターユーザーを使用してゲートウェイデーモンとストレージクラスター間の通信を行いますが、ゲートウェイにはエンドユーザー向けの独自のユーザー管理機能があります。

認証ケイパビリティー

Ceph は、認証されたユーザーが Ceph Monitors および OSD の機能を行使する認可することを示すためにケイパビリティー (capabilities/caps) という用語を使用しています。ケイパビリティーは、プール内のデータやプール内の namespace へのアクセスを制限することもできます。ユーザーを作成または更新する際に、Ceph 管理ユーザーはユーザーのケイパビリティーを設定します。ケイパビリティーの構文は以下の形式に従います。

構文

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

- **Monitor Caps:** モニターのケイパビリティーには、**r**、**w**、**x**、**allow profile CAP**、および **profile rbd** があります。

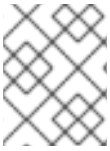
  例

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

- **OSD Caps:** OSD ケイパビリティーには、**r**、**w**、**x**、**class-read**、**class-write**、**profile osd**、**profile rbd**、および **profile rbd-read-only** があります。さらに OSD ケイパビリティーは、プールおよび namespace の設定も許可します。

  構文

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



注記

Ceph Object Gateway デーモン (**radosgw**) は Ceph ストレージクラスターのクライアントであるため、Ceph Storage クラスターデーモンタイプとしては表示されません。

以下のエントリーは、それぞれのケイパビリティーについて説明します。

allow	デーモンのアクセス設定の前に使用してください。
r	ユーザーに読み取り権限を付与します。CRUSH マップを取得するためにモニターで必要です。
w	ユーザーがオブジェクトへの書き込みアクセス権を付与します。
x	クラスメソッド (読み取りおよび書き込みの両方) をユーザーに呼び出し、モニターで **auth** 操作を実行する機能を提供します。

class-read	クラスの読み取りメソッドを呼び出すケイパビリティーを提供します。**x** のサブセット。
class-write	クラスの書き込みメソッドを呼び出すケイパビリティーを提供します。**x** のサブセット。
*	特定のデーモンまたはプールに対する読み取り、書き込み、実行のパーミッション、および admin コマンドの実行権限をユーザーに付与します。
profile osd	OSD として他の OSD またはモニターに接続するためのパーミッションをユーザーに付与します。OSD がレプリケーションのハートビートトラフィックおよびステータス報告を処理できるようにするために OSD に付与されました。
profile bootstrap-osd	OSD のブートストラップ時にキーを追加するパーミッションを持つように、OSD をブートストラップするユーザーパーミッションを付与します。
profile rbd	ユーザーに、Ceph ブロックデバイスへの読み取り/書き込み権限を付与します。
profile rbd-read-only	ユーザーに、Ceph ブロックデバイスへの読み取り専用アクセス権を付与します。

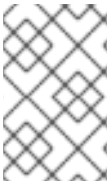
プール

プールは Ceph クライアントのストレージストラテジーを定義し、そのストラテジーの論理パーティションとして機能します。

Ceph デプロイメントでは、さまざまな種類のユースケースをサポートするプールを作成することが一般的です。たとえば、クラウドボリュームまたはイメージ、オブジェクトストレージ、ホットストレージ、コールドストレージなど。OpenStack のバックエンドとして Ceph をデプロイする場合、標準的なデプロイメントにはボリューム、イメージ、バックアップと、**client.glance**、**client.cinder** などユーザーのプールが含まれます。

Namespace

プール内のオブジェクトは、プール内のオブジェクトの論理グループである namespace に関連付けることができます。ユーザーのプールへのアクセスは、ユーザーによる読み書きが名前空間内でのみ行われるように、その名前空間と関連付けることができます。プール内の名前空間に書き込まれたオブジェクトには、その名前空間にアクセスできるユーザーのみがアクセスできます。



注記

現在、名前空間は、**librados** に記述されたアプリケーションにのみ役立ちます。ブロックデバイスやオブジェクトストレージなどの Ceph クライアントでは、この機能は現在サポートされていません。

名前空間の合理的な理由は、各プールが OSD にマッピングされる配置グループのセットを作成するため、プールはユースケース別にデータを分離するために計算量の多い方法になる可能性があるからです。複数のプールが同じ CRUSH 階層とルールセットを使用する場合、OSD のパフォーマンスは負荷の増加に応じて低下する可能性があります。

たとえば、プールには、OSD ごとに約 100 個の配置グループが必要です。そのため、1000 個 の OSD を持つ模範的なクラスターは、1 つのプールに対して 10 万個の配置グループを持つことになります。同じ CRUSH 階層とルールセットにマップされた各プールは、例示的なクラスターにさらに 10 万の配置

グループを作成します。一方、namespace にオブジェクトを書き込むと、別のプールの計算オーバーヘッドを排除して、namespace をオブジェクト名に関連付けられます。ユーザーまたはユーザーセットに個別のプールを作成するのではなく、名前空間を使用できます。

注記

現時点では、**librados** のみを使用できます。

関連情報

- 認証の使用に関する詳細は、Red Hat Ceph Storage 設定ガイド を参照してください。

6.2. CEPH ユーザーの管理

ストレージ管理者は、ユーザーの作成、修正、削除、およびインポートにより Ceph ユーザーを管理できます。Ceph クライアントユーザーは、Ceph クライアントを使用して Red Hat Ceph Storage クラスターデーモンと対話する個人またはアプリケーションのいずれかになります。

6.2.1. Ceph ユーザーのリスト表示

コマンドラインインターフェイスを使用して、ストレージクラスター内のユーザーをリスト表示できます。

前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。
- ノードへのルートレベルのアクセス。

手順

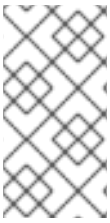
1. ストレージクラスターのユーザーをリスト表示するには、以下を実行します。

   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: AQCgTk9g8sJQNhAAPykcoYUuPc7ljubaFx09HQ==
 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 *
```



注記

ユーザーの **TYPE.ID** 記法が適用され、**osd.0** は **osd** 型のユーザーでその ID は **0**、**client.admin** は **client** 型のユーザーでその ID は **admin**、つまりデフォルトの **client.admin** ユーザーとなります。また、各エントリーには、**key: VALUE** エントリー、および 1 つ以上の **caps:** エントリーがあることに注意してください。

-o FILE_NAME オプションを **ceph auth list** と共に使用して、出力をファイルに保存することができます。

6.2.2. Ceph ユーザー情報の表示

コマンドラインインターフェイスを使用して、Ceph のユーザー情報を表示することができます。

前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。
- ノードへのルートレベルのアクセス。

手順

1. 特定のユーザー、キーおよびケイパビリティーを取得するには、以下を実行します。

   構文

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

   例

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

2. **-o FILE_NAME** オプションを使用することもできます。

   構文

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

   例

   ```
   -
   ```

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

auth export コマンドは、**auth get** と同じですが、エンドユーザーとは無関係な内部 **auid** も出力します。

6.2.3. 新しい Ceph ユーザーの追加

ユーザーを追加すると、ユーザー名 (つまり **TYPE.ID**)、シークレットキー、およびユーザーの作成に使用するコマンドに含まれるケイパビリティー) が作成されます。

ユーザーのキーにより、ユーザーは Ceph Storage クラスターとの認証を行うことができます。ユーザーの機能により、Ceph モニター (**mon**)、Ceph OSD (**osd**)、または Ceph Metadata Server (**mds**) での読み取り、書き込み、実行を承認します。

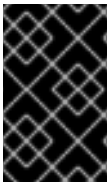
ユーザーを追加する方法はいくつかあります。

- **ceph auth add**: このコマンドは、ユーザーを追加する正規の方法になります。ユーザーを作成し、キーを生成し、指定の機能を追加します。
- **ceph auth get-or-create**: ユーザー名 (括弧内) とキーを持つキーファイルの形式を返すため、このコマンドはユーザーを作成する最も便利な方法です。ユーザーがすでに存在する場合、このコマンドは単にキーファイル形式でユーザー名およびキーを返します。**-o FILE_NAME** オプションを使用して、出力をファイルに保存します。
- **ceph auth get-or-create-key**: このコマンドはユーザーを作成し、ユーザーのキーのみを返す便利な方法です。これは、鍵のみを必要とするクライアント (例: **libvirt**) に役立ちます。ユーザーがすでに存在する場合は、このコマンドが鍵を返すだけです。**-o FILE_NAME** オプションを使用して、出力をファイルに保存します。

クライアントユーザーの作成時に、ケイパビリティーのないユーザーを作成できます。クライアントはモニターからクラスターマップを取得できないため、ケイパビリティーのないユーザーには認証以上のことができません。ただし、後で **ceph auth caps** コマンドを使用してケイパビリティーを追加する場合には、ケイパビリティーがないユーザーを作成することができます。

通常ユーザーは、Ceph OSD における Ceph モニターおよび読み取り/書き込みケイパビリティーにおいて、少なくとも読み取りケイパビリティーを持ちます。また、ユーザーの OSD パーミッションは、多くの場合、特定のプールへのアクセスに制限されます。

```
[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

ユーザーに OSD に対するケイパビリティーを提供する場合に、特定のプールへのアクセスを制限しない場合は、ユーザーはクラスター内のすべてのプールにアクセスできるようになります。

6.2.4. Ceph ユーザーの変更

ceph auth caps コマンドを使用すると、ユーザーを指定してユーザーのケイパビリティーを変更できます。

前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。
- ノードへのルートレベルのアクセス。

手順

1. ケイパビリティーを追加するには、以下の形式を使用します。

   構文

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

   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. ケイパビリティーを削除するには、このケイパビリティーをリセットできます。ユーザーが以前に設定された特定のデーモンにアクセスできないようにするには、空の文字列を指定します。

   例

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

関連情報

- ケイパビリティーの詳細は、**認証ケイパビリティー** を参照してください。

6.2.5. Ceph ユーザーの削除

コマンドラインインターフェイスを使用して、Ceph Storage クラスターからユーザーを削除できます。

前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。
- ノードへのルートレベルのアクセス。

手順

1. ユーザーを削除するには、**ceph auth del** を使用します。

   構文

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

例

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

6.2.6. Ceph ユーザーキーの出力

コマンドラインインターフェイスを使用して、Ceph ユーザーのキー情報を表示することができます。

前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。
- ノードへのルートレベルのアクセス。

手順

- ユーザーの認証キーを標準出力に出力します。

  構文

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

  例

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

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

第7章 CEPH-VOLUME ユーティリティー

ストレージ管理者は、**ceph-volume** ユーティリティーを使用して、Ceph OSD の準備、リスト表示、作成、アクティブ化、非アクティブ化、バッチ処理、トリガー、ザッピング、および移行を行うことができます。**ceph-volume** ユーティリティーは、論理ボリュームを OSD としてデプロイするための単一の目的コマンドラインツールです。プラグインタイプのフレームワークを使用して、異なるデバイス技術を持つ OSD をデプロイします。**ceph-volume** ユーティリティーは、OSD のデプロイに使用する **ceph-disk** ユーティリティーと同様のワークフローに従います。これは、OSD の準備、アクティブ化、および起動を可能にする予測可能で堅牢な方法です。現在、**ceph-volume** ユーティリティーは **lvm** プラグインのみをサポートします。また、今後、その他のテクノロジーをサポートする予定があります。

重要

ceph-disk コマンドは非推奨となりました。

7.1. CEPH ボリュームの LVM プラグイン

LVM タ グを利用することで、**lvm** サブコマンドは OSD に関連付けられたデバイスを照会して保存し、再検出できるため、それらをアクティブにすることができます。これには、**dm-cache** などの lvm ベースのテクノロジーもサポートします。

ceph-volume を使用する場合は、**dm-cache** の使用は透過的になり、**dm-cache** は論理ボリュームのように処理されます。**dm-cache** を使用した場合のパフォーマンスの損益は、特定のワークロードに依存します。一般的に、ランダム読み取りおよび順次読み取りにより、ブロックサイズが小さいとパフォーマンスが向上することになります。ランダムな書き込みや順次書き込みでは、ブロックサイズが大きくなるとパフォーマンスが低下します。

LVM プラグインを使用するには、cephadm シェル内の **ceph-volume** コマンドにサブコマンドとして **lvm** を追加します。

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

以下は、**lvm** のサブコマンドです。

- **prepare** - LVM デバイスをフォーマットし、OSD と関連付けます。
- **activate** - OSD ID に関連付けられた LVM デバイスを検出してマウントし、Ceph OSD を起動します。
- **list** - Ceph に関連する論理ボリュームおよびデバイスをリスト表示します。
- **batch** - 最小限の操作で、マルチ OSD プロビジョニングのためのデバイスを自動的にサイズ調整します。
- **deactivate** - OSD を非アクティブ化します。
- **create** - LVM デバイスから新しい OSD を作成します。
- **trigger** - OSD を起動するための systemd ヘルパー。
- **zap** - 論理ボリュームまたはパーティションからすべてのデータおよびファイルシステムを削除します。
- **migrate** - BlueFS データを別の LVM デバイスに移行します。

- **new-wal** - 指定された論理ボリュームに OSD 用の新しい WAL ボリュームを割り当てます。
- **new-db** - 指定された論理ボリュームに OSD 用の新しい DB ボリュームを割り当てます。

注記

create サブコマンドを使用すると、**prepare** および **activate** サブコマンドが 1 つのサブコマンドに統合されます。

関連情報

- 詳細は、**create** サブコマンドの セクション を参照してください。

7.2. CEPH-VOLUME が CEPH-DISK の代替になる理由

Red Hat Ceph Storage 4 までは、OSD の準備、アクティブ化、作成に **ceph-disk** ユーティリティーが使用されていました。Red Hat Ceph Storage 4 以降では、**ceph-disk** が **ceph-volume** ユーティリティーに置き換えられました。これは、OSD として論理ボリュームをデプロイする 1 つの目的のコマンドラインツールとなることを目的としています。一方、OSD の準備、アクティブ化、および作成時に同様の API を **ceph-disk** に維持します。

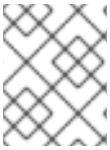
ceph-volume はどのように機能しますか。

ceph-volume は、ハードウェアデバイスのプロビジョニングに関して、現在、レガシーの **ceph-disk** デバイスと LVM (Logical Volume Manager) デバイスの 2 つの方法に対応しているモジュラーツールです。**ceph-volume lvm** コマンドは、LVM タグを使用して、Ceph 固有のデバイスと、OSD との関係に関する情報を保存します。これらのタグを使用して、OSDS に関連付けられたデバイスを後で再検出し、クエリーし、それらをアクティベートできるようにします。LVM および **dm-cache** に基づく技術にも対応しています。

ceph-volume ユーティリティーは **dm-cache** を透過的に使用し、論理ボリュームとして処理します。処理する特定のワークロードに応じて、**dm-cache** を使用した場合のパフォーマンスの損益を考慮するとよいでしょう。一般的に、ブロックサイズが小さくなるとランダムおよび順次読み出し操作のパフォーマンスが向上し、ブロックサイズが大きくなるとランダムおよび連続書き込み操作のパフォーマンスが低下します。**ceph-volume** を使用しても、パフォーマンスが大幅に低下することはありません。

重要

ceph-disk ユーティリティーは非推奨になりました。



注記

ceph-volume simple コマンドは、レガシーの **ceph-disk** デバイスが使用されている場合には、そのデバイスを処理することができます。

ceph-disk はどのように機能しますか。

ceph-disk ユーティリティーは、**upstart** や **sysvinit** のような異なるタイプの init システムを多数サポートしながら、デバイスを検出することができるようにする必要がありました。このため、**ceph-disk** は GUID パーティションテーブル (GPT) パーティションにのみ集中します。具体的には、デバイスを独自の方法でラベル付けする GPT GUID で、次のような質問に答えます。

- このデバイスは **ジャーナル** ですか。

- このデバイスは暗号化されたデータパーティションですか。
- デバイスが部分的に準備されましたか。

これらの質問を解決するために、**ceph-disk** は UDEV ルールを使用して GUID に一致させます。

ceph-disk を使用するデメリットはなんですか。

UDEV ルールを使用して **ceph-disk** を呼び出すと、**ceph-disk systemd** ユニットと **ceph-disk** 実行ファイルの間に行き来が発生することがあります。このプロセスは信頼性が非常に低く、時間がかかるため、ノードのブートプロセス中に OSD が全く起動しなくなることがあります。さらに、UDEV の非同期動作により、これらの問題をデバッグしたり複製することもすることは困難です。

ceph-disk は GPT パーティションと排他的に機能するため、論理ボリュームマネージャー (LVM) ボリュームや同様のデバイスマッパーデバイスなどの他のテクノロジーには対応できません。

GPT パーティションがデバイス検出ワークフローで正しく機能するようにするには、**ceph-disk** で多数の特別なフラグを使用する必要があります。また、これらのパーティションには、デバイスを Ceph が排他的に所有する必要があります。

7.3. CEPH-VOLUME を使用した CEPH OSD の準備

prepare サブコマンドは、OSD バックエンドのオブジェクトストアを準備し、OSD データとジャーナルの両方に論理ボリューム (LV) を消費します。これは、LVM を使用して追加のメタデータタグを追加する以外は、論理ボリュームを変更しません。これらのタグはボリュームの検出を容易にし、Ceph ストレージクラスターの一部としてのボリュームとストレージクラスター内でのボリュームのロールを識別します。

BlueStore OSD バックエンドでは、以下の設定がサポートされます。

- ブロックデバイス、**block.wal** デバイス、および **block.db** デバイス
- ブロックデバイスと **block.wal** デバイス
- ブロックデバイスと **block.db** デバイス
- 1 つのブロックデバイス

prepare サブコマンドは、全デバイスまたはパーティション、または **ブロック** の論理ボリュームを受け入れます。

前提条件

- OSD ノードへのルートレベルのアクセス。
- 必要に応じて、論理ボリュームを作成します。物理デバイスへのパスを指定すると、サブコマンドはデバイスを論理ボリュームに変換します。このアプローチはシンプルですが、論理ボリュームの作成方法を設定したり、変更したりすることはできません。

手順

1. Ceph キーリングを抽出します。

   構文

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

例

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

2. LVM ボリュームを準備します。

構文

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

例

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

a. 必要に応じて、RocksDB 用に別のデバイスを使用する場合は、**--block.db** オプションおよび **--block.wal** オプションを指定します。

構文

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

例

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

b. 必要に応じて、データを暗号化するには、**--dmcrypt** フラグを使用します。

構文

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

例

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

関連情報

- 詳細は、**Red Hat Ceph Storage 管理ガイド** の `ceph-volume` を使用した Ceph OSD のアクティブ化 セクションを参照してください。
- 詳細は、**Red Hat Ceph Storage 管理ガイド**の `ceph-volume` を使用した Ceph OSD の作成 セクションを参照してください。

7.4. CEPH-VOLUME を使用したデバイスのリスト表示

ceph-volume lvm list サブコマンドを使用して、Ceph クラスターに関連付けられた論理ボリュームと

デバイスをリスト表示できます。ただし、その検出を可能にするための十分なメタデータが含まれている必要があります。出力は、デバイスに関連付けられた OSD ID でグループ化されています。論理ボリュームの場合、**devices key** には、論理ボリュームに関連する物理デバイスが入力されます。

場合によっては、**ceph -s** コマンドの出力に次のエラーメッセージが表示されることがあります。

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

このような場合、**ceph device ls-lights** コマンドを使用してデバイスをリスト表示すると、デバイス上のライトに関する詳細が表示されます。情報に基づいて、デバイスのライトを消すことができます。

前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。
- Ceph OSD ノードへのルートレベルのアクセス。

手順

- Ceph クラスター内のデバイスをリスト表示します。

  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
```

- オプション: ストレージクラスター内のデバイスをライト付きでリストします。

  例

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

{
    "fault": [
        "SEAGATE_ST12000NM002G_ZL2KTGCK0000C149"
```

```
    ],
    "ident": []
}
```

a. オプション: デバイスのライトをオフにします。

構文

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

例

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

7.5. CEPH-VOLUME を使用した CEPH OSD のアクティブ化

アクティベーションプロセスにより、システムの起動時に **systemd** ユニットが有効になり、正しい OSD 識別子とその UUID が有効になり、マウントされます。

前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。
- Ceph OSD ノードへのルートレベルのアクセス。
- **ceph-volume** ユーティリティーが準備する Ceph OSD。

手順

1. OSD ノードから OSD ID と OSD FSID を取得します。

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

2. OSD をアクティベートします。

構文

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

例

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

アクティブ化用に準備されているすべての OSD を有効にするには、**--all** オプションを使用します。

例

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

3. オプションで、**trigger** サブコマンドを使用することができます。このコマンドは直接使うことはできず、**systemd** が **ceph-volume lvm activate** への入力をプロキシーするために使用します。これは、systemd とスタートアップから来るメタデータを解析し、OSD に関連する UUID と ID を検出します。

   構文

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

   ここでは、**SYSTEMD_DATA** は、**OSD_ID-OSD_FSID** の形式になります。

   例

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

関連情報

- 詳細は、**Red Hat Ceph Storage 管理ガイド** の **`ceph-volume` を使用した Ceph OSD の準備** セクションを参照してください。
- 詳細は、**Red Hat Ceph Storage 管理ガイド**の **`ceph-volume` を使用した Ceph OSD の作成** セクションを参照してください。

7.6. CEPH-VOLUME を使用した CEPH OSD の非アクティブ化

ceph-volume lvm サブコマンドを使用して、Ceph OSD を非アクティブにすることができます。このサブコマンドは、ボリュームグループと論理ボリュームを削除します。

前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。
- Ceph OSD ノードへのルートレベルのアクセス。
- Ceph OSD は、**ceph-volume** ユーティリティーを使用してアクティブ化されます。

手順

1. OSD ノードから OSD ID を取得します。

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

2. OSD を非アクティブ化します。

   構文

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

   例

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

関連情報

- 詳細は、**Red Hat Ceph Storage 管理ガイド** の `ceph-volume` を使用した Ceph OSD のアクティブ化 セクションを参照してください。
- 詳細は、**Red Hat Ceph Storage 管理ガイド** の `ceph-volume` を使用した Ceph OSD の準備 セクションを参照してください。
- 詳細は、**Red Hat Ceph Storage 管理ガイド**の `ceph-volume` を使用した Ceph OSD の作成 セクションを参照してください。

7.7. CEPH-VOLUME を使用した CEPH OSD の作成

create サブコマンドは **prepare** サブコマンドを呼び出し、**activate** サブコマンドを呼び出します。

前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。
- Ceph OSD ノードへのルートレベルのアクセス。

注記

作成プロセスに対する制御を強化する場合は、サブコマンドの **prepare** および **activate** を個別に使用して、**create** を使用する代わりに OSD を作成できます。この 2 つのサブコマンドを使用すると、大量のデータをリバランスせずに、新規 OSD をストレージクラスターに段階的に導入することができます。**create** サブコマンドを使用すると、完了直後に OSD が **up** および **in** になりますが、どちらのアプローチも同じように機能します。

手順

1. 新規 OSD を作成するには、以下を実行します。

   構文

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

   例

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

関連情報

- 詳細は、**Red Hat Ceph Storage 管理ガイド** の `ceph-volume` を使用した Ceph OSD の準備 セクションを参照してください。
- 詳細は、**Red Hat Ceph Storage 管理ガイド** の `ceph-volume` を使用した Ceph OSD のアクティブ化 セクションを参照してください。

7.8. BLUEFS データの移行

RocksDB データである BlueStore ファイルシステム (BlueFS) のデータは、**migrate** LVM サブコマンドを使用して、ソースボリュームからターゲットボリュームに移行することができます。成功すると、メインボリューム以外のソースボリュームが削除されます。

LVM ボリュームは主にターゲット専用になります。

新しいボリュームは、ソースドライブの 1 つを置き換えて、OSD にアタッチされます。

LVM ボリュームの配置ルールは以下の通りです。

- ソースリストに DB または WAL ボリュームがある場合、ターゲットデバイスはそれを置き換えます。
- ソースリストのボリュームが遅いだけの場合は、**new-db** または **new-wal** コマンドを使用した明示的な割り当てが必要です。

new-db コマンドおよび **new-wal** コマンドは、それぞれ DB または WAL ボリュームとして、特定の論理ボリュームを特定の OSD にアタッチします。

前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。
- Ceph OSD ノードへのルートレベルのアクセス。
- **ceph-volume** ユーティリティーが準備する Ceph OSD。
- ボリュームグループと論理ボリュームが作成されます。

手順

1. **cephadm** シェルにログインします。

   例

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

2. DB または WAL デバイスを追加する OSD を停止します。

   例

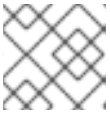
   ```
   [ceph: root@host01 /]# ceph orch daemon stop osd.1
   ```

3. 新しいデバイスをコンテナーにマウントします。

   例

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

4. 特定の論理ボリュームを DB/WAL デバイスとして OSD にアタッチします。



注記

このコマンドは、OSD に DB がアタッチされている場合は失敗します。

構文

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

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. BlueFS のデータ移行は、以下の方法で行うことができます。

- BlueFS データをメインデバイスから DB としてすでにアタッチされている LV に移動します。

  構文

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

  例

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

- BlueFS データを共有のメインデバイスから、新しい DB としてアタッチされる LV に移動します。

  構文

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

  例

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

- BlueFS データを DB デバイスから新しい LV に移動し、DB デバイスを置き換えます。

  構文

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

  例

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

- BlueFS データをメインデバイスおよび DB デバイスから新しい LV に移動し、DB デバイスを置き換えます。

  構文

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

  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
  ```

- BlueFS データをメインデバイス、DB デバイス、および WAL デバイスから新しい LV に移動し、WAL デバイスを削除して、DB デバイスを置き換えます。

  構文

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

  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
  ```

- BlueFS データをメインデバイス、DB デバイス、および WAL デバイスからメインデバイスに移動し、WAL デバイスおよび DB デバイスを削除します。

  構文

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

  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. CEPH-VOLUME でのバッチモードの使用

batch サブコマンドは、単一デバイスが提供されると複数の OSD の作成を自動化します。

ceph-volume コマンドは、ドライブタイプに基づいて OSD の作成に使用する最適な方法を決定します。Ceph OSD の最適化は、利用可能なデバイスによって異なります。

- すべてのデバイスが従来のハードドライブの場合、**batch** はデバイスごとに OSD を 1 つ作成します。

- すべてのデバイスがソリッドステートドライブの場合は、**バッチ** によりデバイスごとに OSD が 2 つ作成されます。
- 従来のハードドライブとソリッドステートドライブが混在している場合、**バッチ** はデータに従来のハードドライブを使用し、ソリッドステートドライブに可能な限り大きいジャーナル (**block.db**) を作成します。

注記

batch サブコマンドは、write-ahead-log (**block.wal**) デバイスに別の論理ボリュームを作成することに対応していません。

前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。
- Ceph OSD ノードへのルートレベルのアクセス。

手順

1. 複数のドライブに OSD を作成するには、以下の手順を実行します。

   構文

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

   例

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

関連情報

- 詳細は、**Red Hat Ceph Storage 管理ガイド**の `ceph-volume` を使用した Ceph OSD の作成 セクションを参照してください。

7.10. CEPH-VOLUME を使用したデータのザッピング

zap サブコマンドは、論理ボリュームまたはパーティションからすべてのデータおよびファイルシステムを削除します。

zap サブコマンドを使用して、再利用のために Ceph OSD で使用される論理ボリューム、パーティション、または raw デバイスをザッピングできます。特定の論理ボリュームまたはパーティションに存在するすべてのファイルシステムが削除され、すべてのデータがパージされます。

オプションで、**--destroy** フラグを使用して、論理ボリューム、パーティション、または物理デバイスを完全に削除することができます。

前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。
- Ceph OSD ノードへのルートレベルのアクセス。

手順

手順

- 論理ボリュームをザッピングします。

  構文

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

  例

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

- パーティションをザッピングします。

  構文

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

  例

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

- raw デバイスをザッピングします。

  構文

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

  例

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

- OSD ID で複数のデバイスをパージします。

  構文

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

  例

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

  注記

  関連デバイスはすべてザッピングされます。

- FSID で OSD をパージします。

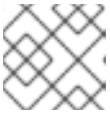
  構文

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

–

例

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



注記

関連デバイスはすべてザッピングされます。

第8章 CEPH パフォーマンスベンチマーク

ストレージ管理者は、Red Hat Ceph Storage クラスターのパフォーマンスをベンチマークできます。本セクションの目的は、Ceph 管理者が Ceph のネイティブベンチマークツールの基本を理解することを目的としています。これらのツールにより、Ceph Storage クラスターの実行方法についての洞察が提供されます。これは、Ceph パフォーマンスベンチマークの最終ガイドではなく、Ceph を適宜調整する方法に関するガイドです。

8.1. パフォーマンスベースライン

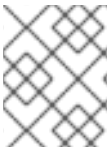
ジャーナル、ディスク、ネットワークのスループットを含む OSD には、比較すべきパフォーマンスベースラインがあるはずです。ベースラインのパフォーマンスデータと Ceph のネイティブツールのデータを比較することで、潜在的なチューニング効果を特定することができます。Red Hat Enterprise Linux には、これらのタスクを実現するために利用可能なオープンソースコミュニティーツールが複数含まれています。

関連情報

- 利用可能なツールの詳細は、ナレッジベースアーティクル Red Hat Enterprise Linux で利用できるベンチマークツールおよびパフォーマンステストツールはありますか？を参照してください。

8.2. CEPH パフォーマンスのベンチマーク

Ceph には、RADOS ストレージクラスターでパフォーマンスベンチマークを行う **rados bench** コマンドが含まれます。このコマンドは、書き込みテストと 2 種類の読み取りテストを実行します。**--no-cleanup** オプションは、読み取りおよび書き込みパフォーマンスの両方をテストする際に使用することが重要です。デフォルトでは、**rados bench** コマンドは、ストレージプールに書き込まれたオブジェクトを削除します。これらのオブジェクトをそのまま残すと、2 つの読み取りテストで、順次読み取りパフォーマンスとランダムな読み取りパフォーマンスを測定できます。



注記

これらのパフォーマンステストを実行する前に、次のコマンドを実行してファイルシステムのキャッシュをすべて破棄します。

例

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

前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。
- ノードへのルートレベルのアクセス。

手順

1. 新しいストレージプールを作成します。

   例

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

2. 新規作成されたストレージプールへの書き込みテストを 10 秒実行します。

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. ストレージプールへの 10 秒間の順次読み取りテストを実行します。

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. ストレージプールに対して、10 秒間ランダムな読み取りテストを実行します。

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. 同時の読み書き数を増やすには、**-t** オプションを使用します (デフォルトは 16 スレッド)。また、**-b** パラメーターは、書き込まれているオブジェクトのサイズを調整することもできます。デフォルトのオブジェクトサイズは 4 MB です。安全な最大オブジェクトサイズは 16 MB です。Red Hat は、このベンチマークテストの複数のコピーを異なるプールで実行することを推奨します。これにより、複数のクライアントのパフォーマンスが変更になりました。
--run-name LABEL オプションを追加して、ベンチマークテスト中に作成するオブジェクトの名前を制御します。実行中の各コマンドインスタンスの **--run-name** ラベルを変更すると、複数の **rados bench** コマンドを同時に実行できます。これにより、複数のクライアントが同じオブジェクトにアクセスしようとし、異なるクライアントが異なるオブジェクトにアクセスしようとすると発生する可能性のある I/O エラーを防ぐことができます。**--run-name** オプションは、実世界のワークロードをシミュレートしようとしているときにも便利です。

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. **rados bench** コマンドで作成したデータを削除します。

   例

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

8.3. CEPH ブロックパフォーマンスのベンチマーク

Ceph には、ブロックデバイスへの順次書き込みをテストする **rbd bench-write** コマンドが含まれます。これは、スループットとレイテンシーの測定を行います。デフォルトのバイトサイズは 4096 で、デフォルトの I/O スレッド数は 16 で、書き込みするデフォルトのバイト数は 1 GB です。これらのデフォルトは、それぞれ **--io-size** オプション、**--io-threads** オプション、および **--io-total** オプションで変更できます。

前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。
- ノードへのルートレベルのアクセス。

手順

- ブロックデバイスに対して書き込みパフォーマンステストを実行します

  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
..
```

関連情報

- **rbd** コマンドの詳細は、**Red Hat Ceph Storage ブロックデバイスガイド** の Ceph ブロックデバイス の章を参照してください。

第9章 CEPH パフォーマンスカウンター

ストレージ管理者は、Red Hat Ceph Storage クラスターのパフォーマンスメトリクスを収集できます。Ceph パフォーマンスカウンターは、内部インフラストラクチャーメトリクスのコレクションです。このメトリクスデータの収集、集計、およびグラフ化は、さまざまなツールで実行でき、パフォーマンス分析に役立ちます。

9.1. CEPH パフォーマンスカウンターへのアクセス

パフォーマンスカウンターは、Ceph Monitor および OSD のソケットインターフェイスを介して利用できます。各デーモンのソケットファイルは、デフォルトでは **/var/run/ceph** の下にあります。パフォーマンスカウンターは、コレクション名にグループ化されます。これらのコレクション名はサブシステムまたはサブシステムのインスタンスを表します。

以下は、Monitor および OSD コレクション名のカテゴリーのリストです。それぞれの簡単な説明を以下に示します。

コレクション名カテゴリーの監視

- Cluster Metrics - ストレージクラスターに関する情報を表示します (モニター、OSD、プール、PG)。
- Level Database Metrics - バックエンドの **KeyValueStore** データベースに関する情報を表示します。
- Monitor Metrics - 一般的なモニター情報を表示します。
- Paxos Metrics - クラスタークォーラム管理に関する情報を表示します。
- Throttle Metrics - モニターのスロットリング方法の統計を表示します。

OSD コレクションの名前カテゴリー

- Write Back Throttle Metrics - 書き込みバックスロットルがフラッシュされていない IO を追跡する方法についての統計を表示します。
- Level Database Metrics - バックエンドの **KeyValueStore** データベースに関する情報を表示します。
- Objecter Metrics - さまざまなオブジェクトベースの操作に関する情報を表示します。
- Read and Write Operations Metrics - さまざまな読み取りおよび書き込み操作に関する情報を表示します。
- Recovery State Metrics - さまざまなリカバリーの状態のレイテンシーを表示します。
- OSD Throttle Metrics - OSD のスロットリング方法の統計の表示

RADOS ゲートウェイコレクションの名前カテゴリー

- Object Gateway Client Metrics - GET 要求および PUT 要求の統計を表示します。
- Objecter Metrics - さまざまなオブジェクトベースの操作に関する情報を表示します。
- Object Gateway Throttle Metrics: OSD のスロットリングに関する統計の表示

9.2. CEPH パフォーマンスカウンターの表示

ceph daemon DAEMON_NAME perf schema コマンドは、利用可能なメトリクスを出力します。各メトリクスには、関連付けられたビットフィールド値タイプがあります。

前提条件

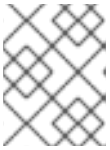
- 稼働中の Red Hat Ceph Storage クラスターがある。
- ノードへのルートレベルのアクセス。

手順

1. メトリクスのスキーマを表示するには、以下を実行します。

   構文

   ```
   ceph daemon DAEMON_NAME perf schema
   ```

   注記

   デーモンを実行するノードから **ceph daemon** コマンドを実行する必要があります。

2. モニターノードから **ceph daemon DAEMON_NAME perf schema** コマンドを実行するには、以下を実行します。

   例

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

3. OSD ノードから **ceph daemon DAEMON_NAME perf schema** コマンドを実行するには、以下を実行します。

   例

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

表9.1 ビットフィールド値の定義

ビット	意味
1	**浮動小数点数**
2	**署名されていない 64 ビットの整数値**
4	**平均 (合計 + カウント)**
8	**カウンター**

各値には、型 (浮動小数点または整数値) を示すビット 1 または 2 が設定されます。ビット 4 が設定されている場合、読み取る値は合計とカウントの 2 つになります。ビット 8 が設定されている場合、以前の間隔の平均は、以前の読み取り以降、合計差分になります。これは、カウントデルタで除算されます。値を除算すると、有効期間の平均値が提供されることになります。通常、これらはレイテンシー、リクエスト数、およびリクエストレイテンシーの合計を測定するために使用されます。ビットの値は組み合わせられます (例: 5、6、10)。ビット値 5 は、ビット 1 とビット 4 の組み合わせです。つまり、平均は浮動小数点の値になります。ビット値 6 は、ビット 2 とビット 4 の組み合わせです。これは、平均値が整数になることを意味します。ビット値 10 は、ビット 2 とビット 8 の組み合わせです。これは、カウンター値が整数値であることを意味します。

関連情報

- 詳細は、**Red Hat Ceph Storage 管理ガイド**の **平均数と合計** セクションを参照してください。

9.3. CEPH パフォーマンスカウンターのダンプ

ceph daemon .. perf dump コマンドは、現在の値を出力し、各サブシステムのコレクション名でメトリクスをグループ化します。

前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。
- ノードへのルートレベルのアクセス。

手順

1. 現在のメトリクスデータを表示するには、以下を実行します。

   構文

   ```
   ceph daemon DAEMON_NAME perf dump
   ```

   注記

   デーモンを実行するノードから **ceph daemon** コマンドを実行する必要があります。

2. Monitor ノードから **ceph daemon .. perf dump** コマンドを実行するには、以下のコマンドを実行します。

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

3. OSD ノードから **ceph daemon .. perf dump** コマンドを実行するには、以下のコマンドを実行します。

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

関連情報

- 利用可能な各 Monitor メトリクスの簡単な説明を表示するには、**Ceph Monitor メトリクスの表** を参照してください。

9.4. 平均数と合計

すべてのレイテンシー番号は、ビットフィールドの値は 5 です。このフィールドには、平均数と合計の浮動小数点値が含まれます。**avgcount** は、この範囲内の操作数で、**sum** はレイテンシーの合計 (秒単位) です。**sum** を **avgcount** で除算すると、操作ごとのレイテンシーを把握することができます。

関連情報

- 利用可能な各 OSD メトリクスの簡単な説明を表示するには、Ceph OSD メトリクスの表 を参照してください。

9.5. CEPH MONITOR メトリクス

- クラスターメトリクステーブル
- レベルのデータベースメトリクステーブル
- 一般的なモニターメトリクステーブル
- Paxos Metrics テーブル
- スロットルメトリクステーブル

表9.2 クラスターメトリクステーブル

コレクション名	メトリクス名	ビットフィールド値	簡単な説明
cluster	**num_mon**	2	モニター数
	num_mon_quorum	2	クォーラムのモニター数
	num_osd	2	OSD の合計数
	num_osd_up	2	稼働中の OSD 数
	num_osd_in	2	クラスターにある OSD 数
	osd_epoch	2	OSD マップの現在のエポック
	osd_bytes	2	クラスターの合計容量 (バイト単位)
	osd_bytes_used	2	クラスターで使用されているバイト数
	osd_bytes_avail	2	クラスターで利用可能なバイト数
	num_pool	2	プールの数
	num_pg	2	配置グループの合計数

コレクション名	メトリクス名	ビットフィールド値	簡単な説明
	num_pg_active_clean	2	active+clean 状態の配置グループの数
	num_pg_active	2	アクティブな状態の配置グループの数
	num_pg_peering	2	ピア状態の配置グループの数
	num_object	2	クラスター上のオブジェクトの合計数
	num_object_degraded	2	パフォーマンス低下 (レプリカが欠落している) オブジェクトの数
	num_object_misplaced	2	配置が間違っているオブジェクトの数 (クラスター内の間違った場所)
	num_object_unfound	2	不明なオブジェクトの数
	num_bytes	2	すべてのオブジェクトの合計バイト数
	num_mds_up	2	稼働している MDS の数
	num_mds_in	2	クラスターにある MDS の数
	num_mds_failed	2	失敗した MDS の数
	mds_epoch	2	MDS マップの現在のエポック

表9.3 レベルのデータベースメトリクステーブル

コレクション名	メトリクス名	ビットフィールド値	簡単な説明
leveldb	**leveldb_get**	10	取得
	leveldb_transaction	10	トランザクション
	leveldb_compact	10	補完
	leveldb_compact_range	10	範囲ごとの比較

コレクション名	メトリクス名	ビットフィールド値	簡単な説明
	leveldb_compact_queue_merge	10	圧縮キューにおける範囲のマージ
	leveldb_compact_queue_len	2	圧縮キューの長さ

表9.4 一般的なモニターメトリクステーブル

コレクション名	メトリクス名	ビットフィールド値	簡単な説明
mon	**num_sessions**	2	現在開いているモニターセッションの数
	session_add	10	作成されたモニターセッションの数
	session_rm	10	モニターにおける remove_session 呼び出しの数
	session_trim	10	トリミングされたモニターセッションの数
	num_elections	10	参加する選択モニターの数
	election_call	10	モニターにより開始した選択の数
	election_win	10	モニターが勝利した選択の数
	election_lose	10	モニターにより失われた選択の数

表9.5 Paxos Metrics テーブル

コレクション名	メトリクス名	ビットフィールド値	簡単な説明
paxos	**start_leader**	10	リーダーロールで始まります。
	start_peon	10	peon ロールで開始します。
	restart	10	再起動
	refresh	10	更新
	refresh_latency	5	更新の待ち時間

コレクション名	メトリクス名	ビットフィールド値	簡単な説明
	begin	10	開始および処理開始
	begin_keys	6	開始時のトランザクションのキー
	begin_bytes	6	トランザクションの開始時にデータ
	begin_latency	5	開始操作のレイテンシー
	commit	10	コミット
	commit_keys	6	コミット時にトランザクションのキー
	commit_bytes	6	コミット時のトランザクションのデータ
	commit_latency	5	コミットレイテンシー
	collect	10	Peon が収集する
	collect_keys	6	peon collect 上のトランザクションのキー
	collect_bytes	6	peon collect 上のトランザクションのデータ
	collect_latency	5	Peon がレイテンシーを収集
	collect_uncommitted	10	開始および処理された収集のコミットされていない値
	collect_timeout	10	タイムアウトの収集
	accept_timeout	10	タイムアウトの受け入れ
	lease_ack_timeout	10	リースの確認タイムアウト
	lease_timeout	10	リースタイムアウト
	store_state	10	共有状態をディスクに保存

コレクション名	メトリクス名	ビットフィールド値	簡単な説明
	store_state_keys	6	保存された状態のトランザクションのキー
	store_state_bytes	6	保存された状態のトランザクションのデータ
	store_state_latency	5	状態レイテンシーの保存
	share_state	10	状態の共有
	share_state_keys	6	共有状態のキー
	share_state_bytes	6	共有状態のデータ
	new_pn	10	新しい提案番号のクエリー
	new_pn_latency	5	レイテンシーを取得する新しい提案番号

表9.6 スロットルメトリクステーブル

コレクション名	メトリクス名	ビットフィールド値	簡単な説明
throttle-*	**val**	10	現在利用できるスロットル
	max	10	スロットルの最大値
	get	10	取得
	get_sum	10	取得したデータ
	get_or_fail_fail	10	get_or_fail 時にブロックされる
	get_or_fail_success	10	get_or_fail 時の get 成功
	take	10	取得
	take_sum	10	取得したデータ
	put	10	送る
	put_sum	10	データを送る
	wait	5	待機レイテンシー

9.6. CEPH OSD メトリクス

- ライトバックスロットルメトリクステーブル
- レベルのデータベースメトリクステーブル
- Objecter Metrics テーブル
- 読み出し操作および書き込み操作のメトリクステーブル
- リカバリー状態のメトリクステーブル
- OSD スロットルのメトリクステーブル

表9.7 ライトバックスロットルメトリクステーブル

コレクション名	メトリクス名	ビットフィールド値	簡単な説明
WBThrottle	**bytes_dirtied**	2	ダーティーデータ
	bytes_wb	2	書き込まれたデータ
	ios_dirtied	2	ダーティー操作
	ios_wb	2	書き込みされた操作
	inodes_dirtied	2	書き込みを待っているエントリー
	inodes_wb	2	書き込まれたエントリー

表9.8 レベルのデータベースメトリクステーブル

コレクション名	メトリクス名	ビットフィールド値	簡単な説明
leveldb	**leveldb_get**	10	取得
	leveldb_transaction	10	トランザクション
	leveldb_compact	10	補完
	leveldb_compact_range	10	範囲ごとの比較
	leveldb_compact_queue_merge	10	圧縮キューにおける範囲のマージ
	leveldb_compact_queue_len	2	圧縮キューの長さ

表9.9 Objecter Metrics テーブル

コレクション名	メトリクス名	ビットフィールド値	簡単な説明
objecter	**op_active**	2	アクティブな操作
	op_laggy	2	遅延操作
	op_send	10	送信された操作
	op_send_bytes	10	送信されたデータ
	op_resend	10	再送信捜査
	op_ack	10	コールバックのコミット
	op_commit	10	操作のコミット
	op	10	操作
	op_r	10	読み取り操作
	op_w	10	書き込み操作
	op_rmw	10	read-modify-write 操作
	op_pg	10	PG 操作
	osdop_stat	10	統計操作
	osdop_create	10	オブジェクト操作の作成
	osdop_read	10	読み取り操作
	osdop_write	10	書き込み操作
	osdop_writefull	10	完全なオブジェクト操作の書き込み
	osdop_append	10	追加操作
	osdop_zero	10	オブジェクトをゼロ操作に設定
	osdop_truncate	10	オブジェクト操作の切り捨て
	osdop_delete	10	オブジェクト操作の削除

コレクション名	メトリクス名	ビットフィールド値	簡単な説明
	osdop_mapext	10	エクステント操作のマップ
	osdop_sparse_read	10	スパース読み取り操作
	osdop_clonerange	10	範囲のクローン操作
	osdop_getxattr	10	xattr 操作の取得
	osdop_setxattr	10	xattr 操作の設定
	osdop_cmpxattr	10	xattr の比較操作
	osdop_rmxattr	10	xattr 操作の削除
	osdop_resetxattrs	10	xattr 操作のリセット
	osdop_tmap_up	10	TMAP 更新操作
	osdop_tmap_put	10	TMAP の put 操作
	osdop_tmap_get	10	TMAP の get 操作
	osdop_call	10	操作の呼び出し (実行)
	osdop_watch	10	オブジェクト操作による監視
	osdop_notify	10	オブジェクト操作に関する通知
	osdop_src_cmpxattr	10	複数演算における拡張属性比較
	osdop_other	10	その他の操作
	linger_active	2	アクティブな linger 操作
	linger_send	10	送信された linger 操作
	linger_resend	10	再送信された linger 操作
	linger_ping	10	linger 操作に ping が送信
	poolop_active	2	アクティブなプール操作
	poolop_send	10	送信したプール操作
	poolop_resend	10	再送されたプール操作

コレクション名	メトリクス名	ビットフィールド値	簡単な説明
	poolstat_active	2	アクティブな get pool stat 操作
	poolstat_send	10	送信されたプール統計操作
	poolstat_resend	10	再送信されたプール統計
	statfs_active	2	statfs 操作
	statfs_send	10	送信された FS 統計
	statfs_resend	10	再送信された FS 統計
	command_active	2	アクティブなコマンド
	command_send	10	送信されたコマンド
	command_resend	10	再送信された コマンド
	map_epoch	2	OSD マップエポック
	map_full	10	受け取った完全な OSD マップ
	map_inc	10	受け取った増分 OSD マップ
	osd_sessions	2	セッションを開く
	osd_session_open	10	開いたセッション
	osd_session_close	10	閉じたセッション
	osd_laggy	2	Laggy OSD セッション

表9.10 読み出し操作および書き込み操作のメトリクステーブル

コレクション名	メトリクス名	ビットフィールド値	簡単な説明
osd	**op_wip**	2	現在処理中のレプリケーション操作 (プライマリー)
	op_in_bytes	10	クライアント操作の合計書き込みサイズ
	op_out_bytes	10	クライアント操作合計読み取りサイズ

コレクション名	メトリクス名	ビットフィールド値	簡単な説明
	op_latency	5	クライアント操作のレイテンシー (キュー時間を含む)
	op_process_latency	5	クライアント操作のレイテンシー (キュー時間を除く)
	op_r	10	クライアントの読み取り操作
	op_r_out_bytes	10	読み込まれたクライアントデータ
	op_r_latency	5	読み取り操作のレイテンシー (キュー時間を含む)
	op_r_process_latency	5	読み取り操作のレイテンシー (キュー時間を除く)
	op_w	10	クライアントの書き込み操作
	op_w_in_bytes	10	書き込まれたクライアントデータ
	op_w_rlat	5	クライアントの書き込み操作の読み取り可能/適用されるレイテンシー
	op_w_latency	5	書き込み操作のレイテンシー (キュー時間を含む)
	op_w_process_latency	5	書き込み操作のレイテンシー (キュー時間を除く)
	op_rw	10	クライアントの read-modify-write 操作
	op_rw_in_bytes	10	クライアントの read-modify-write 操作の書き込み
	op_rw_out_bytes	10	クライアントの read-modify-write 操作の読み出し
	op_rw_rlat	5	クライアントの読み取り/書き込み操作の読み取り/適用のレイテンシー
	op_rw_latency	5	read-modify-write 操作のレイテンシー (キュー時間を含む)

コレクション名	メトリクス名	ビットフィールド値	簡単な説明
	op_rw_process_latency	5	read-modify-write 操作のレイテンシー (キュー時間を除く)
	subop	10	サブ操作
	subop_in_bytes	10	サブ操作の合計サイズ
	subop_latency	5	サブ操作レイテンシー
	subop_w	10	レプリケートされた書き込み
	subop_w_in_bytes	10	複製された書き込みデータのサイズ
	subop_w_latency	5	レプリケートされた書き込みレイテンシー
	subop_pull	10	サブオペレーションのプルリクエスト
	subop_pull_latency	5	サブオペレーションのプルレイテンシー
	subop_push	10	サブオペレーションのプッシュメッセージ
	subop_push_in_bytes	10	サブオペレーションのプッシュサイズ
	subop_push_latency	5	サブオペレーションのプッシュレイテンシー
	pull	10	送信されたプル要求
	push	10	送信されたメッセージをプッシュ
	push_out_bytes	10	プッシュされたサイズ
	push_in	10	インバウンドプッシュメッセージ
	push_in_bytes	10	インバウンドプッシュされるサイズ
	recovery_ops	10	開始したリカバリー操作

コレクション名	メトリクス名	ビットフィールド値	簡単な説明
	loadavg	2	CPU 負荷
	buffer_bytes	2	割り当てられたバッファー合計サイズ
	numpg	2	配置グループ
	numpg_primary	2	この osd がプライマリーとなる配置グループ
	numpg_replica	2	この osd がレプリカである配置グループ
	numpg_stray	2	この osd から削除する準備ができている配置グループ
	heartbeat_to_peers	2	送信先のハートビート (ping) ピア
	heartbeat_from_peers	2	受信元ハートビート (ping) のピア
	map_messages	10	OSD マップメッセージ
	map_message_epochs	10	OSD マップエポック
	map_message_epoch_dups	10	OSD マップの複製
	stat_bytes	2	OSD のサイズ
	stat_bytes_used	2	使用されている領域
	stat_bytes_avail	2	利用可能な領域
	copyfrom	10	RADOS の copy-from 操作
	tier_promote	10	階層の昇格
	tier_flush	10	階層フラッシュ
	tier_flush_fail	10	レイヤーフラッシュの失敗
	tier_try_flush	10	階層のフラッシュ試行

コレクション名	メトリクス名	ビットフィールド値	簡単な説明
	tier_try_flush_fail	10	階層のフラッシュ試行の失敗
	tier_evict	10	階層エビクション
	tier_whiteout	10	階層のホワイトアウト
	tier_dirty	10	ダーティー階層フラグセット
	tier_clean	10	消去されたダーティー階層フラグ
	tier_delay	10	階層遅延 (エージェント待機)
	tier_proxy_read	10	階層プロキシーの読み込み
	agent_wake	10	階層化エージェントのウェイクアップ
	agent_skip	10	エージェントによりスキップされるオブジェクト
	agent_flush	10	階層化エージェントのフラッシュ
	agent_evict	10	階層化エージェントエビクション
	object_ctx_cache_hit	10	オブジェクトコンテキストキャッシュのヒット数
	object_ctx_cache_total	10	オブジェクトコンテキストキャッシュの検索
	ceph_cluster_osd_blocklist_count	2	ブロックリストに登録されたクライアントの数

表9.11 リカバリー状態のメトリクステーブル

コレクション名	メトリクス名	ビットフィールド値	簡単な説明
recoverystate_perf	**initial_latency**	5	リカバリーの状態の最初のレイテンシー
	started_latency	5	リカバリー状態のレイテンシーを開始

コレクション名	メトリクス名	ビットフィールド値	簡単な説明
	reset_latency	5	リカバリー状態レイテンシーのリセット
	start_latency	5	リカバリー状態レイテンシーの開始
	primary_latency	5	プライマリーリカバリーの状態のレイテンシー
	peering_latency	5	リカバリー状態のレイテンシーのピア設定
	backfilling_latency	5	リカバリー状態レイテンシーのバックフィル
	waitremotebackfillreserved_latency	5	リモートバックフィルの予約されたリカバリー状態レイテンシーを待機する
	waitlocalbackfillreserved_latency	5	ローカルバックフィルの予約されたリカバリー状態のレイテンシーを待機する
	notbackfilling_latency	5	Notbackfilling のリカバリー状態のレイテンシー
	repnotrecovering_latency	5	Repnotrecovering リカバリー状態のレイテンシー
	repwaitrecoveryreserved_latency	5	repwaitrecovery が予約したリカバリー状態のレイテンシー
	repwaitbackfillreserved_latency	5	repwaitbackfill が予約したリカバリー状態のレイテンシー
	RepRecovering_latency	5	repRecovering リカバリー状態のレイテンシー
	activating_latency	5	リカバリー状態のレイテンシーの有効化
	waitlocalrecoveryreserved_latency	5	waitlocalrecovery が予約したリカバリー状態のレイテンシー
	waitremoterecoveryreserved_latency	5	waitremoterecovery が予約したリカバリー状態のレイテンシー

コレクション名	メトリクス名	ビットフィールド値	簡単な説明
	recovering_latency	5	リカバリー状態のレイテンシーのリカバリー
	recovered_latency	5	リカバリーしたリカバリー状態のレイテンシー
	clean_latency	5	リカバリー状態のレイテンシーの削除
	active_latency	5	アクティブリカバリーの状態のレイテンシー
	replicaactive_latency	5	replicaactive のリカバリー状態レイテンシー
	stray_latency	5	迷子のリカバリー状態レイテンシー
	getinfo_latency	5	getinfo リカバリー状態レイテンシー
	getlog_latency	5	getlog リカバリー状態レイテンシー
	waitactingchange_latency	5	Waitactingchange リカバリー状態レイテンシー
	incomplete_latency	5	不完全なリカバリー状態レイテンシー
	getmissing_latency	5	復旧状態のレイテンシーの取得
	waitupthru_latency	5	waitupthru リカバリー状態のレイテンシー

表9.12 OSD スロットルのメトリクステーブル

コレクション名	メトリクス名	ビットフィールド値	簡単な説明
throttle-*	**val**	10	現在利用できるスロットル
	max	10	スロットルの最大値
	get	10	取得
	get_sum	10	取得したデータ

コレクション名	メトリクス名	ビットフィールド値	簡単な説明
	get_or_fail_fail	10	get_or_fail 時にブロックされる
	get_or_fail_success	10	get_or_fail 時の get 成功
	take	10	取得
	take_sum	10	取得したデータ
	put	10	送る
	put_sum	10	データを送る
	wait	5	待機レイテンシー

9.7. CEPH OBJECT GATEWAY メトリクス

- Ceph Object Gateway クライアントテーブル
- Objecter Metrics テーブル
- Ceph Object Gateway スロットルメトリクステーブル

表9.13 Ceph Object Gateway クライアントメトリクステーブル

コレクション名	メトリクス名	ビットフィールド値	簡単な説明
client.rgw.<rgw_node_name>	**req**	10	要求
	failed_req	10	中止された要求
	copy_obj_ops	10	オブジェクトのコピー
	copy_obj_bytes	10	オブジェクトのコピーのサイズ
	copy_obj_lat	10	オブジェクトのコピーのレイテンシー
	del_obj_ops	10	オブジェクトの削除
	del_obj_bytes	10	オブジェクトの削除のサイズ
	del_obj_lat	10	オブジェクトの削除のレイテンシー

コレクション名	メトリクス名	ビットフィールド値	簡単な説明
	del_bucket_ops	10	バケットの削除
	del_bucket_lat	10	バケットの削除のレイテンシー
	get	10	取得
	get_b	10	取得サイズ
	get_initial_lat	5	取得レイテンシー
	list_obj_ops	10	オブジェクトのリスト表示
	list_obj_lat	10	オブジェクトのリスト表示のレイテンシー
	list_buckets_ops	10	バケットのリスト表示
	list_buckets_lat	10	バケットのリスト表示のレイテンシー
	put	10	送る
	put_b	10	送信サイズ
	put_initial_lat	5	レイテンシーの送信
	qlen	2	キューの長さ
	qactive	2	アクティブなリクエストキュー
	cache_hit	10	キャッシュのヒット数
	cache_miss	10	キャッシュミス
	keystone_token_cache_hit	10	Keystone トークンキャッシュのヒット数
	keystone_token_cache_miss	10	Keystone のトークンキャッシュのミス

表9.14 Objecter Metrics テーブル

コレクション名	メトリクス名	ビットフィールド値	簡単な説明
objecter	**op_active**	2	アクティブな操作

コレクション名	メトリクス名	ビットフィールド値	簡単な説明
	op_laggy	2	遅延操作
	op_send	10	送信された操作
	op_send_bytes	10	送信されたデータ
	op_resend	10	再送信捜査
	op_ack	10	コールバックのコミット
	op_commit	10	操作のコミット
	op	10	操作
	op_r	10	読み取り操作
	op_w	10	書き込み操作
	op_rmw	10	read-modify-write 操作
	op_pg	10	PG 操作
	osdop_stat	10	統計操作
	osdop_create	10	オブジェクト操作の作成
	osdop_read	10	読み取り操作
	osdop_write	10	書き込み操作
	osdop_writefull	10	完全なオブジェクト操作の書き込み
	osdop_append	10	追加操作
	osdop_zero	10	オブジェクトをゼロ操作に設定
	osdop_truncate	10	オブジェクト操作の切り捨て
	osdop_delete	10	オブジェクト操作の削除
	osdop_mapext	10	エクステント操作のマップ
	osdop_sparse_read	10	スパース読み取り操作

コレクション名	メトリクス名	ビットフィールド値	簡単な説明
	osdop_clonerange	10	範囲のクローン操作
	osdop_getxattr	10	xattr 操作の取得
	osdop_setxattr	10	xattr 操作の設定
	osdop_cmpxattr	10	xattr の比較操作
	osdop_rmxattr	10	xattr 操作の削除
	osdop_resetxattrs	10	xattr 操作のリセット
	osdop_tmap_up	10	TMAP 更新操作
	osdop_tmap_put	10	TMAP の put 操作
	osdop_tmap_get	10	TMAP の get 操作
	osdop_call	10	操作の呼び出し (実行)
	osdop_watch	10	オブジェクト操作による監視
	osdop_notify	10	オブジェクト操作に関する通知
	osdop_src_cmpxattr	10	複数演算における拡張属性比較
	osdop_other	10	その他の操作
	linger_active	2	アクティブな linger 操作
	linger_send	10	送信された linger 操作
	linger_resend	10	再送信された linger 操作
	linger_ping	10	linger 操作に ping が送信
	poolop_active	2	アクティブなプール操作
	poolop_send	10	送信したプール操作
	poolop_resend	10	再送されたプール操作
	poolstat_active	2	アクティブな get pool stat 操作

コレクション名	メトリクス名	ビットフィールド値	簡単な説明
	poolstat_send	10	送信されたプール統計操作
	poolstat_resend	10	再送信されたプール統計
	statfs_active	2	statfs 操作
	statfs_send	10	送信された FS 統計
	statfs_resend	10	再送信された FS 統計
	command_active	2	アクティブなコマンド
	command_send	10	送信されたコマンド
	command_resend	10	再送信された コマンド
	map_epoch	2	OSD マップエポック
	map_full	10	受け取った完全な OSD マップ
	map_inc	10	受け取った増分 OSD マップ
	osd_sessions	2	セッションを開く
	osd_session_open	10	開いたセッション
	osd_session_close	10	閉じたセッション
	osd_laggy	2	Laggy OSD セッション

表9.15 Ceph Object Gateway スロットルメトリクステーブル

コレクション名	メトリクス名	ビットフィールド値	簡単な説明
throttle-*	**val**	10	現在利用できるスロットル
	max	10	スロットルの最大値
	get	10	取得
	get_sum	10	取得したデータ
	get_or_fail_fail	10	get_or_fail 時にブロックされる

コレクション名	メトリクス名	ビットフィールド値	簡単な説明
	get_or_fail_success	10	get_or_fail 時の get 成功
	take	10	取得
	take_sum	10	取得したデータ
	put	10	送る
	put_sum	10	データを送る
	wait	5	待機レイテンシー

第10章 MCLOCK OSD スケジューラー

ストレージ管理者は、mClock キューイングスケジューラーを使用して Red Hat Ceph Storage のサービス品質 (QoS) を実装できます。これは、dmClock と呼ばれる mClock アルゴリズムの適応に基づいています。

mClock OSD スケジューラーは、設定プロファイルを使用して適切な予約、重み付け、制限タグをサービスタイプに割り当てることで、必要な QoS を提供します。

mClock OSD スケジューラーは、OSD の IOPS 機能 (自動的に決定) と最大シーケンシャル帯域幅機能を使用して、さまざまなデバイスタイプ (SSD または HDD) の QoS 計算を実行します (**m Clock 設定オプション** セクションの **osd_m Clock_max_sequential_bandwidth_hdd** および **osd_m Clock_max_sequential_bandwidth_ssd** を参照)。

10.1. MCLOCK OSD スケジューラーと WPQ OSD スケジューラーの比較

mClock OSD スケジューラーは、Weighted Priority Queue (WPQ) OSD スケジューラーに代わり、Red Hat Ceph Storage 6.1 のデフォルトスケジューラーになりました。

重要

mClock スケジューラーは、BlueStore OSD でサポートされています。

mClock OSD スケジューラーには、即時キューがあります。このキューには、即時レスポンスを必要とする操作が追加されます。即時キューは mClock によって処理されず、単純な先入れ先出しキューとして機能し、最優先されます。

OSD レプリケーション操作、OSD 操作応答、ピアリング、最優先としてマークされたリカバリーなどの操作は、即時キューに入れられます。他のすべての操作は、mClock アルゴリズムに従って機能する mClock キューに追加されます。

mClock キュー **mclock_scheduler** は、操作が属しているバケット (**pg recovery**、**pg scrub**、**snap trim**、**client op**、**pg deletion**) に基づき操作に優先順位を付けます。

バックグラウンド操作が進行中の場合、mClock プロファイルを使用すると、WPQ スケジューラーと比較して平均クライアントスループット (つまり 1 秒あたりの入出力操作 (IOPS)) は大幅に高く、レイテンシーは低くなります。これは、mClock が QoS パラメーターを効果的に割り当てるからです。

関連情報

- 詳細は、**mClock プロファイル** のセクションを参照してください。

10.2. 入出力リソースの割り当て

このセクションでは、QoS 制御が予約、制限、重みの割り当てを実行して、内部でどのように機能するかについて説明します。QoS 制御は、mClock プロファイルによって自動的に設定されるため、ユーザーが設定する必要はありません。制御のチューニングは、利用可能な mClock プロファイルを使用してのみ実行できます。

dmClock アルゴリズムは、重みに比例して Ceph クラスターの入出力 (I/O) リソースを割り当てます。サービスがリソースに対して公平に競合できるように、予約の下限と制限の上限を設定します。

現在、**mclock_scheduler** 操作キューは、I/O リソースが含まれる Ceph サービスを以下のバケットに分割します。

- **client op**: クライアントによって発行された 1 秒あたりの入出力操作数 (IOPS)。
- **pg deletion**: プライマリー Ceph OSD によって発行された IOPS。
- **snap trim**: スナップショットのトリミングに関連するリクエスト。
- **pg recovery**: リカバリー関連のリクエスト。
- **pg Scrub**: スクラブ関連のリクエスト。

リソースは、次の 3 つのタグセットを使用して分割されます。つまり、各サービスタイプの共有は、これらの 3 つのタグによって制御されます。

- 予約
- 制限
- 重み

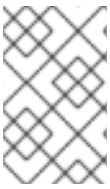
予約

サービスに割り当てられた最小 IOPS。サービスの予約が多いほど、必要に応じて、より多くのリソースを所有することが保証されます。

たとえば、予約が 0.1 (または 10%) に設定されているサービスには、常に OSD の IOPS 容量の 10% がそれ自体に割り当てられます。そのため、クライアントが大量の I/O リクエストを発行し始めても、すべての I/O リソースが枯渇することはなく、高負荷のクラスターでもサービスの動作が枯渇することはありません。

制限

サービスに割り当てられた IOPS の上限。サービスは、それが必要なリクエストで、競合するサービスがない場合でも、設定された 1 秒あたりの対応数を超えてリクエストを取得することはありません。サービスが適用された制限を超えると、制限が復元されるまで操作は操作キューに残ります。



注記

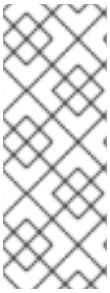
値が **0** (無効) に設定されている場合、サービスは制限設定による制限を受けず、他に競合する操作がない場合はすべてのリソースを使用できます。これは、mClock プロファイルでは MAX として表されます。

注記

予約および制限パラメーターの割り当ては、Ceph OSD の下で、バッキングデバイスのタイプ (HDD または SSD) に基づきシャードごとに行われます。**osd_op_num_shards_hdd** および **osd_op_num_shards_ssd** パラメーターの詳細は、OSD オブジェクトストレージデーモンの設定オプション を参照してください。

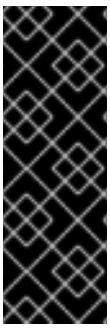
重み

追加容量またはシステムが不十分な場合における容量の配分比。サービスの重みが競合するサービスよりも大きい場合、サービスは I/O リソースを多く使用できます。



注記

サービスの予約値と制限値は、OSD の合計 IOPS 容量の割合に基づいて指定されます。この割合は、mClock プロファイルではパーセンテージとして表されます。重量にはありません。重みは相対的であるため、あるクラスのリクエストの重みが 9 で、別のクラスの重みが 1 である場合、リクエストは 9 対 1 の比率で実行されます。ただし、これは予約が満たされた場合にのみ実行され、その値には予約フェーズで実行された操作が含まれます。



重要

重みが **W** に設定されている場合、指定されたクラスのリクエストに対して、次に入力される重みタグは **1/W** と前の重みタグ、または現在時刻のいずれか大きい方になります。つまり、**W** が大きく、そのために **1/W** が小さくなりすぎる場合、算出されるタグは現在時刻の値になり、割り当てられることはありません。

したがって、重みの値は、1 秒あたりに処理されると予想されるリクエストの数より小さくなければなりません。

10.3. MCLOCK 操作キューに影響を与える要因

Red Hat Ceph Storage 内の mClock 操作キューに影響を与える要因は 3 つあります。

- クライアント操作のためのシャード数
- 操作シーケンサー内の操作数
- Ceph OSD の分散システムの使用

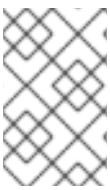
クライアント操作のためのシャード数

Ceph OSD へのリクエストは、配置グループ識別子によりシャード化されます。各シャードには独自の mClock キューがあり、これらのキューは相互に作用したり、シャード間で情報を共有したりしません。

シャードの数は、次の設定オプションで制御できます。

- **osd_op_num_shards**
- **osd_op_num_shards_hdd**
- **osd_op_num_shards_ssd**

シャードの数が少ないと、mClock キューの影響が大きくなりますが、他の悪影響が生じる可能性があります。



注記

設定オプション (**osd_op_num_shards**、**osd_op_num_shards_hdd**、**osd_op_num_shards_ssd**) で定義されているデフォルトのシャード数を使用します。

操作シーケンサー内の操作数

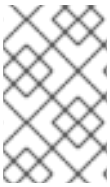
リクエストは操作キューから操作シーケンサーに転送され、そこで処理されます。mClock スケジューラーは操作キューにあります。これは、操作シーケンサーに転送するオペレーションを決定します。

操作シーケンサーで許可される操作の数は複雑な問題です。ここでの目標は、操作シーケンサーに十分な操作を確保し、ディスクやネットワークアクセスが他の操作を完了するまで待機している間も、常に操作を実行できるようにすることです。

ただし、mClock は、操作シーケンサーに転送される操作を制御できなくなりました。そのため、mClock の影響を最大化するためには、操作シーケンサー内の操作を可能な限り少なくすることを目標とする必要があります。

操作シーケンサーの操作数に影響を与える設定オプションは次のとおりです。

- **bluestore_throttle_bytes**
- **bluestore_throttle_deferred_bytes**
- **bluestore_throttle_cost_per_io**
- **bluestore_throttle_cost_per_io_hdd**
- **bluestore_throttle_cost_per_io_ssd**



注記

bluestore_throttle_bytes および **bluestore_throttle_deferred_bytes** オプションで定義されているデフォルト値を使用します。ただし、これらのオプションはベンチマークフェーズで決定できます。

Ceph OSD の分散システムの使用

mClock アルゴリズムの影響を左右する 3 つ目の要因は、複数の Ceph OSD に対してリクエストが行われ、各 Ceph OSD が複数のシャードを持つことができる分散システムの使用です。ただし、現行の Red Hat Ceph Storage はmClock アルゴリズムを使用しており、これは mClock の分散バージョンではありません。

注記

dmClock は、mClock の分散バージョンです。

関連情報

- **osd_op_num_shards_hdd** および **osd_op_num_shards_ssd** パラメーターの詳細については、Object Storage Daemon (OSD) の設定オプションを参照してください。
- BlueStore スロットルパラメーターの詳細は、BlueStore の設定オプション を参照してください。
- 詳細は、OSD の手動ベンチマーク を参照してください。

10.4. MCLOCK の設定

mClock を直感的に使いやすくするために、Red Hat Ceph Storage 6 では mClock 設定プロファイルが導入されました。mClock プロファイルでは、下位ベルの詳細をユーザーに表示しないため、mClock の設定と使用が容易になります。

mClock プロファイルでサービス品質 (QoS) 関連のパラメーターを設定するには、次の入力パラメーターが必要です。

- 各 Ceph OSD における 1 秒あたりの入出力操作 (IOPS) の合計容量。これは自動的に決定されます。
- 各 OS の最大連続帯域幅容量 (MiB/秒)。**osd_m Clock_max_sequential_bandwidth_hdd/ssd** オプションを参照してください。
- 有効にする mClock プロファイルのタイプ。デフォルトは **balanced** です。

Ceph OSD は、指定されたプロファイルの設定を使用して、下位レベルの mClock および Ceph パラメーターを決定し、適用します。mClock プロファイルがパラメーターを適用することで、クライアント I/O と OSD のバックグラウンド操作の間で QoS を調整できます。

関連情報

- OSD 容量の自動決定の詳細は **Ceph OSD 容量の決定** を参照してください。

10.5. MCLOCK クライアント

mClock スケジューラーは、さまざまなタイプの Ceph サービスからのリクエストを処理します。mClock は、各サービスをクライアントのタイプと見なします。処理されるリクエストのタイプに応じて、mClock クライアントは次のバケットに分類されます。

- クライアント - Ceph の外部クライアントによって発行された入出力 (I/O) リクエストを処理します。
- バックグラウンド回復 - 内部回復要求を処理します。
- バックグラウンドのベストエフォート - 内部バックフィル、スクラブ、スナップトリム、および配置グループ (PG) の削除リクエストを処理します。

mClock スケジューラーは、QoS 計算に使用される操作のコストを **osd_mclock_max_capacity_iops_hdd** | **osd_mclock_max_capacity_iops_ssd**、**osd_mclock_max_sequential_bandwidth_hdd** | **osd_mclock_max_sequential_bandwidth_ssd** および **osd_op_num_shards_hdd** | **osd_op_num_shards_ssd** パラメーターから導出します。

10.6. MCLOCK プロファイル

mClock プロファイルは構成設定です。実行中の Red Hat Ceph Storage クラスターに適用すると、バックグラウンドリカバリー、**scrub**、**snap trim**、**client op**、**pg deletion** などの各種クライアントクラスに属する IOPS 操作のスロットリングが可能になります。

mClock プロファイルは、容量制限とユーザーが選択した mClock プロファイルタイプを使用して、下位レベルの mClock リソース制御設定パラメーターを決定し、それらを透過的に適用します。その他の Red Hat Ceph Storage 設定パラメーターも適用されます。下位レベルの mClock リソース制御パラメーターとは、リソース共有を制御する予約、制限、重みです。mClock プロファイルは、クライアントタイプごとにこれらのパラメーターを割り当てます。

10.6.1. mClock プロファイルのタイプ

mClock プロファイルは、**ビルトイン** プロファイルと **カスタム** プロファイルに分類できます。

いずれかの mClock プロファイルがアクティブな場合、次の Red Hat Ceph Storage 設定のスリープオプションは **0** に設定され、無効になります。

- **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**

これは、mClock スケジューラーが操作キューから次の操作を選択して操作シーケンサーに転送するタイミングを決定できるようにするためです。その結果、すべてのクライアントで意図した QoS が提供されます。

カスタム プロファイル

このプロファイルにより、ユーザーはすべての mClock 設定パラメーターを完全に制御できます。これは注意して使用する必要があり、mClock および Red Hat Ceph Storage 関連の設定オプションを理解している上級ユーザーを対象としています。

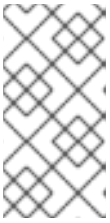
ビルトイン プロファイル

ビルトイン プロファイルが有効になっている場合、mClock スケジューラーは、クライアントタイプごとに有効なプロファイルに基づき、下位レベルの mClock パラメーター (つまり予約、重み、制限) を計算します。

mClock パラメーターは、事前に提供された Ceph OSD の最大容量に基づいて計算されます。そのため、ビルトインプロファイルを使用している場合、以下の mClock 設定オプションは変更できません。

- **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**



注記

これらのデフォルトは、**config set**、**config daemon**、**config Tell** コマンドなどの config サブシステムコマンドを使用して変更することはできません。上記のコマンドは成功を報告しますが、m Clock QoS パラメーターはそれぞれの組み込みプロファイルのデフォルトに戻ります。

次のリカバリーおよびバックフィル関連の Ceph オプションは、mClock のデフォルトにオーバーライドされます。

警告

組み込みプロファイルはこれらのオプションに基づいて最適化されるため、これらのオプションを変更しないでください。これらのデフォルトを変更すると、予期しないパフォーマンス結果が生じる可能性があります。

- **osd_max_backfills**
- **osd_recovery_max_active**
- **osd_recovery_max_active_hdd**
- **osd_recovery_max_active_ssd**

次のオプションは、フォアグラウンドクライアント操作のパフォーマンスを最大化するための現在のデフォルトと同じ mClock のデフォルトを示します。

osd_max_backfills

元のデフォルト

1

mClock のデフォルト

1

osd_recovery_max_active

元のデフォルト

0

mClock のデフォルト

0

osd_recovery_max_active_hdd

元のデフォルト

3

mClock のデフォルト

3

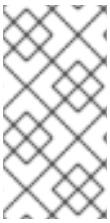
osd_recovery_max_active_sdd

元のデフォルト

10

mClock のデフォルト

10



注記

上記の mClock のデフォルトは、デフォルトで **false** に設定されている **osd_mClock_override_recovery_settings** を有効にすることで、必要な場合にのみ変更できます。これらのパラメーターを変更する場合は、**バックフィルおよびリカバリーオプションの変更** を参照してください。

ビルトイン プロファイルタイプ

ユーザーは、次の **built-in** プロファイルタイプから選択できます。

- **balanced** (デフォルト)
- **high_client_ops**
- **high_recovery_ops**

注記

以下のリストに記載されている値は、サービスタイプに割り当てられた Ceph OSD の合計 IOPS 容量の割合を表します。

- **balanced**:

デフォルトの mClock プロファイルは、クライアント IO とリカバリー IO の優先順位の妥協点を表すため、**balanced** に設定されています。クライアント操作とバックグラウンド回復操作に同等の予約または優先度を割り当てます。バックグラウンドのベストエフォート操作には低い予約が与えられるため、競合する操作がある場合には完了までに時間がかかります。このプロファイルは、外部クライアントのパフォーマンス要件が重要ではなく、OSD 内で注意が必要な他のバックグラウンド操作がある場合の、クラスターの通常または定常状態の要件を満たします。

場合によっては、クライアント操作または回復操作のいずれかに高い優先順位を与える必要がある場合があります。このような要件を満たすには、クライアント IO を優先する **high_client_ops** プロファイル、またはリカバリー IO を優先する **high_recovery_ops** プロファイルのいずれかを選択できます。これらのプロファイルについては、以下でさらに説明します。

サービスの種類: クライアント

予約

50%

制限

最大

重み

1

サービスの種類: バックグラウンドリカバリー

予約

50%

制限

最大

重み

1

サービスタイプ: バックグラウンドベストエフォート

予約

最小

制限

90%

重み

1

- **high_client_ops**

このプロファイルは、Ceph OSD のバックグラウンド操作と比較して、より多くの予約と制限をクライアント操作に割り当てることにより、バックグラウンドアクティビティーでのクライアントパフォーマンスを最適化します。たとえば、このプロファイルを有効にすると、回復が遅くなる代わりに、I/O 集中型のアプリケーションに必要なパフォーマンスを持続的に提供できます。以下のリストは、プロファイルによって設定されるリソース制御パラメーターを示しています。

サービスの種類: クライアント

予約

60%

制限

最大

重み

2

サービスの種類: バックグラウンドリカバリー

予約

40%

制限

最大

重み

1

サービスタイプ: バックグラウンドベストエフォート

予約

最小

制限

70%

重み

1

- **high_recovery_ops**

このプロファイルは、外部クライアントや Ceph OSD 内の他のバックグラウンド操作と比較して、バックグラウンドリカバリーパフォーマンスを最適化します。

たとえば、管理者が一時的に有効にして、非ピーク時間中にバックグラウンド回復を加速することができます。以下のリストは、プロファイルによって設定されるリソース制御パラメーターを示しています。

サービスの種類: クライアント

予約

30%

制限

最大

重み

1

サービスの種類: バックグラウンドリカバリー

予約

70%

制限

最大

重み

2

サービスタイプ: バックグラウンドベストエフォート

予約

最小

制限

最大

重み

1

関連情報

- mClock 設定オプションの詳細は **mClock 設定オプション** を参照してください。

10.6.2. mClock プロファイルの変更

デフォルトの mClock プロファイルは **Balanced** に設定されています。**組み込み** プロファイルの他のタイプは、**high_client_ops** および **high_recovery_ops** です。

注記

上級ユーザーでない限り、**カスタム** プロファイルは推奨しません。

前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。
- Ceph Monitor ホストへの root レベルのアクセス。

手順

1. Cephadm シェルにログインします。

   例

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

2. **osd_mclock_profile** オプションを設定します。

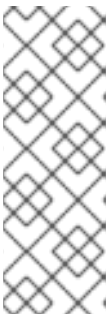
   構文

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

   例

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

   この例では、プロファイルを変更して、**osd.0** でのリカバリーを高速化します。



注記

最適なパフォーマンスを得るには、次のコマンドを使用して、すべての Ceph OSD でプロファイルを設定する必要があります。

構文

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

10.6.3. ビルトイン プロファイルと カスタム プロファイルの切り替え

次の手順では、**ビルトイン** プロファイルから **カスタム** プロファイル、およびその逆の切り替えについて説明します。

すべての mClock 設定オプションを完全に制御したい場合は、**カスタム** プロファイルに切り替えます。ただし、**カスタム** プロファイルの使用は、上級ユーザーにのみ推奨しています。

前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。
- Ceph Monitor ホストへの root レベルのアクセス。

ビルトイン プロファイルから カスタム プロファイルへの切り替え

1. Cephadm シェルにログインします。

   例

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

2. **カスタム** プロファイルに切り替えます。

   構文

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

   例

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

   注記

   最適なパフォーマンスを得るには、次のコマンドを使用して、すべての Ceph OSD でプロファイルを設定する必要があります。

   例

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

3. オプション: **カスタム** プロファイルに切り替えた後、mClock 設定オプションを変更します。

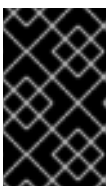
   構文

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

   例

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

   この例では、特定の OSD **osd.0** のクライアント予約 IOPS 比を 0.5 (50%) に変更します。

   重要

   予約の合計が OSD の IOPS 容量の最大割合 (1.0) を超えないように、バックグラウンドリカバリーやバックグラウンドベストエフォートなどの他のサービスの予約を適宜変更します。

カスタム プロファイルから ビルトイン プロファイルへの切り替え

1. cephadm シェルにログインします。

   例

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

2. **ビルトイン** プロファイルを設定します。

   構文

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

   例

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

   この例では、すべての Ceph OSD で **ビルトイン** プロファイルを **high_client_ops** に設定します。

3. データベース内の既存のカスタム mClock 設定を決定します。

   例

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

4. 前の手順で決定したカスタム mClock 設定を削除します。

   構文

   ```
   ceph config rm osd MCLOCK_CONFIGURATION_OPTION
   ```

   例

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

   この例では、すべての Ceph OSD で設定された設定オプション **osd_mclock_scheduler_client_res** を削除します。

   すべての既存のカスタム mClock 設定がセントラル設定データベースから削除されると、**high_client_ops** に関連する設定が適用されます。

5. Ceph OSD の設定を確認します。

   構文

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

   例

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

関連情報

- **ビルトイン** プロファイルで変更できない mClock 設定オプションのリストについては、mClock プロファイルのタイプ を参照してください。

10.6.4. mClock プロファイルの一時的な切り替え

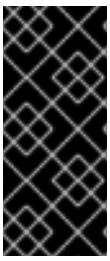
このセクションでは、mClock プロファイルを一時的に切り替える手順を説明しています。

警告

これは、上級ユーザー向け、または実験的なテスト用のセクションです。予期しない結果が生じる可能性があるため、実行中のストレージクラスターでは以下のコマンドを使用しないでください。

注記

以下のコマンドを使用した Ceph OSD の設定変更は一時的なものであり、Ceph OSD を再起動すると失われます。



重要

このセクションで説明されている、コマンドを使用してオーバーライドされる設定オプションは、**ceph config set osd.OSD_ID** コマンドを使用しても変更できません。変更は、Ceph OSD が再起動されるまで有効になりません。これは、設定サブシステムの設計に基づく意図的なものです。ただし、これらのコマンドを使用して一時的に変更するこさらに変更を加えることができます。

前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。
- Ceph Monitor ホストへの root レベルのアクセス。

手順

1. Cephadm シェルにログインします。

   例

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

2. 次のコマンドを実行して、mClock 設定をオーバーライドします。

   構文

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

   例

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

この例では、**osd.0** の **osd_mclock_profile** オプションがオーバーライドされます。

3. オプション: 前述の **ceph tell osd.OSD_ID injectargs** の代わりに使用できます。

構文

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

例

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

注記

上記のコマンドを使用して、**カスタム** プロファイルの個々の QoS 関連の設定オプションを一時的に変更することもできます。

10.6.5. mClock プロファイルを使用した劣化および置き忘れたオブジェクトの回復率

劣化したオブジェクトのリカバリーは、バックグラウンドリカバリーバケットに分類されます。すべての mClock プロファイルにおいて、劣化したオブジェクトのリカバリーは、置き忘れられたオブジェクトのリカバリーよりも高い優先順位が与えられます。これは、劣化したオブジェクトには、単なる置き忘れられたオブジェクトには存在しないデータの安全性の問題が存在するためです。

バックフィルまたは置き忘れたオブジェクトの回復操作は、バックグラウンドのベストエフォートバケットに分類されます。**Balanced** および **high_client_ops** mClock プロファイルによれば、バックグラウンドベストエフォートクライアントは予約 (ゼロに設定) によって制約されませんが、他に競合するサービスがない場合、参加している OSD の容量の一部を使用するように制限されます。

したがって、**balanced** または **high_client_ops** プロファイルを使用し、他のバックグラウンドで競合するサービスがアクティブな場合、以前の **WeightedPriorityQueue** (WPQ) スケジューラーと比較すると、バックフィル速度が遅くなることが予想されます。

より高いバックフィル率が必要な場合は、以下のセクションに記載されている手順に従ってください。

埋め戻し率の向上

Balanced または **high_client_ops** プロファイルを使用するときにバックフィル速度を速くするには、次の手順に従います。

- バックフィルの間、high_recovery_opsmClock プロファイルに切り替えます。これを実現するには、**mClock プロファイルの変更を** 参照してください。バックフィルフェーズが完了したら、mClock プロファイルを以前にアクティブだったプロファイルに切り替えます。high_recovery_ops プロファイルを使用してもバックフィル率に大幅な改善が見られない場合は、次のステップに進みます。
- mClock プロファイルを以前にアクティブだったプロファイルに戻します。
- osd_max_backfills をより高い値 (たとえば、**3)** に変更します。これを実現するには、**バックフィルとリカバリーオプションの変更** を参照してください。

- バックフィルが完了したら、手順 3 で説明したのと同じ手順に従って、osd_max_backfills をデフォルト値の 1 にリセットできます。

警告

osd_max_backfills を変更すると、他の操作が発生する可能性があることに注意してください。たとえば、バックフィルフェーズ中にクライアント操作の遅延が長くなる可能性があります。したがって、クラスター内の他の操作へのパフォーマンスへの影響を最小限に抑えるために、**osd_max_backfills** を少しずつ増やすことを推奨します。

10.6.6. backfills および recovery オプションの変更

ceph config set コマンドを使用して、**backfills** および **recovery** オプションを変更します。

変更可能なバックフィルまたはリカバリーオプションは、mClock プロファイルのタイプ に一覧表示されています。

警告

これは、上級ユーザー向け、または実験的なテスト用のセクションです。予期しない結果が生じる可能性があるため、実行中のストレージクラスターでは以下のコマンドを使用しないでください。

実験的なテスト場合、クラスターが値を処理できない場合、またはデフォルト設定でパフォーマンスが低下する場合にのみ値を変更してください。

重要

mClock のデフォルトのバックフィルまたはリカバリーオプションの変更は、**osd_mclock_override_recovery_settings** オプションによって制限され、デフォルトで **false** に設定されています。

osd_mclock_override_recovery_settings を **true** に設定せずにデフォルトのバックフィルまたはリカバリーオプションを変更しようとすると、オプションは mClock のデフォルトにリセットされ、クラスターログに警告メッセージが記録されます。

前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。
- Ceph Monitor ホストへの root レベルのアクセス。

手順

1. Cephadm シェルにログインします。

例

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

2. すべての Ceph OSD で **osd_mclock_override_recovery_settings** 設定オプションを **true** に設定します。

例

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

3. **backfills** または **recovery** オプションを設定します。

構文

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

例

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

4. 数秒待ってから、特定の OSD の設定を確認します。

構文

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

例

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

5. すべての OSD で **osd_mclock_override_recovery_settings** 設定オプションを **false** にリセットします。

例

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

10.7. CEPH OSD 容量の決定

合計 IOPS に関する Ceph OSD 容量は、Ceph OSD の初期化中に自動的に決定されます。これは、Ceph OSD ベンチツールを実行し、デバイスタイプに応じて **osd_mclock_max_capacity_iops_hdd、ssd** オプションのデフォルト値をオーバーライドすることによって行われます。Ceph OSD 容量を設定するために、ユーザーがその他の操作や入力を行う必要はありません。

自動化された手順により決定された非現実的な Ceph OSD 容量に対する処理

特定の条件では、Ceph OSD ベンチツールは、ドライブの設定やその他の環境関連の条件に基づき、非現実的または誇張された結果を示す場合があります。

この非現実的な容量がパフォーマンスに及ぼす影響を軽減するために、OSD デバイスタイプに応じていくつかのしきい値設定オプションが定義および使用されます。

- **osd_mclock_iops_capacity_threshold_hdd** = 500
- **osd_mclock_iops_capacity_threshold_ssd** = 80000

これらのパラメーターは、次のコマンドを実行して確認できます。

```
[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
```

注記

OSD を手動でベンチマークするか、BlueStore スロットルパラメーターを手動で調整する場合は、**OSD の手動ベンチマーク** を参照してください。

クラスターが起動した後、次のコマンドを実行して OSD の容量を確認できます。

構文

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

例

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

上記の例では、基盤となるデバイスが SSD である Red Hat Ceph Storage ノード上の **osd.0** の最大容量を表示できます。

次の自動化された手順が実行されます。

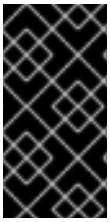
デフォルトの OSD 容量の使用へのフォールバック

Ceph OSD ベンチツールが上記のしきい値を超える測定値を報告した場合、フォールバックメカニズムにより **osd_mclock_max_capacity_iops_hdd** または **osd_mclock_max_capacity_iops_ssd** のデフォルト値に戻ります。しきい値設定オプションは、使用するドライブのタイプに基づき再設定できます。

測定値がしきい値を超えた場合、クラスター警告がログに記録されます。

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

デフォルトの容量が Ceph OSD の容量を正確に示していない場合は、ドライブで Fio などのツールを使用してカスタムベンチマークを実行し、**最大 OSD 容量の指定** で説明されているとおり **osd_mclock_max_capacity_iops_hdd、ssd** オプションをオーバーライドすることを強く推奨します。

関連情報

- Ceph OSD を手動でベンチマークするか、BlueStore スロットルパラメーターを手動で調整する場合は、**手動による OSD ベンチマーク** を参照してください。
- **osd_mclock_max_capacity_iops_hdd、ssd** および **osd_mclock_iops_capacity_threshold_hdd、ssd** オプションの詳細については、**mClock 設定オプション** を参照してください。

10.7.1. OSD 容量の確認

ストレージクラスターをセットアップした後、Ceph OSD の容量を確認できます。

前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。
- Ceph Monitor ホストへの root レベルのアクセス。

手順

1. Cephadm シェルにログインします。

   例

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

2. Ceph OSD の容量を確認します。

   構文

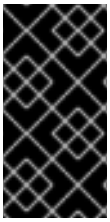
   ```
   ceph config show osd.OSD_ID osd_mclock_max_capacity_iops_[hdd, ssd]
   ```

   例

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

   21500.000000
   ```

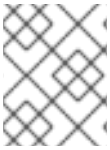
10.7.2. OSD の手動ベンチマーク

Ceph OSD を手動でベンチマークするには、既存のベンチマークツール (Fio など) を使用できます。使用するツールまたはコマンドに関係なく、以下の手順を実行します。



重要

シャードの数と BlueStore スロットルパラメーターは、mClock 操作キューに影響を与えます。したがって、mclock スケジューラーの影響を最大化するには、これらの値を慎重に設定することが重要です。これらの値の詳細については、**mClock 操作キューに影響を与える要因** を参照してください。



注記

このセクションで説明している手順は、OSD 初期化時に自動的に決定された Ceph OSD 容量をオーバーライドする場合にのみ必要です。

注記

ベンチマークデータをすでに決定しており、Ceph OSD の最大 OSD 容量を手動でオーバーライドする場合は、**最大 OSD 容量の指定** セクションに移動してください。

前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。
- Ceph Monitor ホストへの root レベルのアクセス。

手順

1. Cephadm シェルにログインします。

   例

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

2. Ceph OSD をベンチマークします。

   構文

   ```
   ceph tell osd.OSD_ID bench [TOTAL_BYTES] [BYTES_PER_WRITE] [OBJ_SIZE] [NUM_OBJS]
   ```

   ここでは、以下のようになります。

   - **TOTAL_BYTES**: 書き込みバイト数の合計。
   - **BYTES_PER_WRITE**: 書き込みごとのブロックサイズ。
   - **OBJ_SIZE**: オブジェクトごとのバイト数。
   - **NUM_OBJS** : 書き込むオブジェクトの数。

   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. 正しい BlueStore スロットル値の決定

このセクションはオプションであり、正しい BlueStore スロットル値を決定するための手順を詳しく説明しています。この手順では、デフォルトのシャードを使用します。

重要

テストを実行する前に、正確な測定値を取得するためにキャッシュをクリアします。次のコマンドを使用して、各ベンチマーク実行の間に OSD キャッシュをクリアします。

構文

```
ceph tell osd.OSD_ID cache drop
```

例

```
[ceph: root@host01 /]# ceph tell osd.0 cache drop
```

前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。
- ベンチマークする OSD をホストする Ceph Monitor ノードへのルートレベルのアクセス。

手順

1. Cephadm シェルにログインします。

   例

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

2. OSD で単純な 4KiB ランダム書き込みワークロードを実行します。

   構文

   ```
   ceph tell osd.OSD_ID bench 12288000 4096 4194304 100
   ```

   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 **osd bench** コマンドの出力から得た全体的なスループット。この値は、デフォルトの BlueStore スロットルオプションが有効な場合のベースラインスループットです。

3. 前のコマンドの出力から得た全体的なスループット (つまり IOPS) をメモします。
4. お使いの環境の BlueStore スロットル値を決定することが目的の場合は、**bluestore_throttle_bytes** および **bluestore_throttle_deferred_bytes** オプションを 32 KiB (32768 バイト) に設定します。

   構文

   ```
   ceph config set osd.OSD_ID bluestore_throttle_bytes 32768
   ceph config set osd.OSD_ID bluestore_throttle_deferred_bytes 32768
   ```

   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
   ```

   それ以外の場合は、次のセクション (**OSD の最大容量の指定**) に進みます。
5. OSD ベンチコマンドを使用して、前と同じように 4KiB ランダム書き込みテストを実行します。

   例

   ```
   [ceph: root@host01 /]# ceph tell osd.0 bench 12288000 4096 4194304 100
   ```

6. 出力からの全体的なスループットに注目し、その値を前の手順でメモしたベースラインスループットと比較します。
7. スループットがベースラインと一致しない場合は、BlueStore スロットルオプションを 2 倍の値に増加させます。
8. 得られたスループットがベースライン値に非常に近くなるまで、4KiB のランダム書き込みテストを実行し、値をベースラインスループットと比較して BlueStore スロットルオプションを 2 倍の値に増加させる手順を繰り返します。

注記

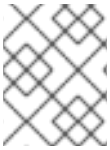
たとえば、NVMe SSD を搭載したマシンでのベンチマーク実行時に、mClock の影響を最大化するために、BlueStore スロットルと遅延バイトの両方の値が 256 KiB に決定されました。HDD の場合、全体的なスループットがベースラインスループットとほぼ同じになる値は 40 MiB でした。

通常、HDD の場合、BlueStore スロットル値は SSD と比較して高くなることが予想されます。

10.7.4. 最大 OSD 容量の指定

OSD の初期化中に自動的に設定された Ceph OSD の最大容量をオーバーライドできます。

これらの手順はオプションです。デフォルト容量が Ceph OSD の容量を正確に示していない場合は、次の手順を実行します。



注記

OSD の手動ベンチマーク で説明されているとおり、最初にベンチマークデータを決定してください。

前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。
- Ceph Monitor ホストへの root レベルのアクセス。

手順

1. Cephadm シェルにログインします。

   例

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

2. OSD の **sd_mclock_max_capacity_iops_[hdd, ssd]** オプションを設定します。

   構文

   ```
   ceph config set osd.OSD_ID osd_mclock_max_capacity_iops_[hdd,ssd] VALUE
   ```

   例

   ```
   [ceph: root@host01 /]# ceph config set osd.0 osd_mclock_max_capacity_iops_hdd 350
   ```

   この例では、基になるデバイスタイプが HDD である **osd.0** の最大容量を 350 IOPS に設定します。

第11章 BLUESTORE

BlueStore は OSD デーモンのバックエンドオブジェクトストアであり、オブジェクトをブロックデバイスに直接配置します。

重要

BlueStore は、本番環境で OSD デーモン向けに高パフォーマンスのバックエンドを提供します。デフォルトでは、BlueStore はセルフチューニングするように設定されています。BlueStore を手動でチューニングした方が環境のパフォーマンスが良いと判断された場合は、Red Hat サポート に連絡して設定の詳細を共有し、自動チューニングのケイパビリティーを改善する支援を受けるようにしてください。Red Hat は、フィードバックをお待ちしており、お客様の提案に感謝いたします。

11.1. CEPH BLUESTORE

以下は、BlueStore を使用する主な機能の一部です。

ストレージデバイスの直接管理

BlueStore は raw ブロックデバイスまたはパーティションを使用します。これにより、XFS などのローカルファイルシステムなど、抽象化の層が回避され、パフォーマンスの制限や複雑さの増加が発生する可能性があります。

RocksDB を使用したメタデータ管理

BlueStore は、ディスク上の場所をブロックするオブジェクト名からのマッピングなど、内部メタデータの管理に RocksDB のキーと値データベースを使用します。

完全なデータおよびメタデータのチェックサム

デフォルトでは、BlueStore に書き込まれたすべてのデータおよびメタデータは、1 つ以上のチェックサムによって保護されます。検証せずにディスクから読み取られたり、ユーザーに返されたデータやメタデータはありません。

インライン圧縮

データは、ディスクに書き込まれる前に圧縮できます。

効率的なコピーオンライト

Ceph Block Device および Ceph File System のスナップショットは、BlueStore に効率的に実装されるコピーオンライトのクローンメカニズムに依存します。これにより、通常のスナップショットと、効率的な 2 フェーズコミットを実装するためにクローン作成に依存するイレイジャーコーディングプールの両方で効率的な I/O が実現します。

大きな二重書き込みなし

BlueStore は、まずブロックデバイス上の未割り当ての領域に新しいデータを書き込み、次にディスクの新しい領域を参照するためにオブジェクトのメタデータを更新する RocksDB トランザクションをコミットします。書き込み操作が設定可能なサイズしきい値を下回る場合にのみ、書き込み優先ジャーナリング方式にフォールバックします。

マルチデバイスのサポート

BlueStore は、複数のブロックデバイスを使用して異なるデータを保存できます。たとえば、データ用のハードディスクドライブ (HDD)、メタデータ用のソリッドステートドライブ (SSD)、不揮発性メモリー (NVM) や不揮発性ランダムアクセスメモリー (NVRAM)、RocksDB のライトアヘッドログ (WAL) 用の永続メモリーなどです。詳細は、Ceph BlueStore デバイス を参照してください。

ブロックデバイスの効率的な使用方法

BlueStore はファイルシステムを使用しないため、ストレージデバイスキャッシュを削除する必要が最小限に抑えられます。

割り当てメタデータ

割り当てメタデータは RocksDB のスタンドアロンオブジェクトを使用しなくなりました。割り当て情報は、RocksDB にすでに保存されているシステム内の全 onode の集約割り当て状態から推測できます。BlueStore V3 コードは、割り当て時に RocksDB の更新をスキップし、**umount** 中にすべての OSD 割り当て状態でアロケーターオブジェクトの完全なステージを実行します。これにより、IOPS が 25% になり、ランダム書き込みのワークロードが小さくなり、レイテンシーが短縮されます。ただし、割り当てメタデータを再作成するためにすべての onode を繰り返し処理する必要があるため、通常はリカバリー時間が数分長くなります。

キャッシュの age bin

Red Hat Ceph Storage は、さまざまなキャッシュ内の項目を「Age Bin」に関連付け、すべてのキャッシュ項目の想定的な有効期限が表示されます。

11.2. CEPH BLUESTORE デバイス

BlueStore は、バックエンドの 1 つ、2 つ、または 3 つのストレージデバイスを管理します。

- プライマリー
- WAL
- DB

最も単純なケースでは、BlueStore は単一のプライマリーストレージデバイスを使用します。通常、ストレージデバイスは全体として使用され、BlueStore によって直接管理されるデバイス全体を占有します。プライマリーデバイスは、data ディレクトリーの **block** シンボリックリンクで識別されます。

データディレクトリーは **tmpfs** マウントであり、OSD に関する情報 (識別子、所属するクラスター、その秘密キーリングなど) を保持するすべての一般的な OSD ファイルが読み込まれます。

ストレージデバイスは、以下を含む 2 つの部分に分割されます。

- **OSD メタデータ**: OSD の基本的なメタデータが含まれる XFS でフォーマットされた小規模なパーティション。このデータディレクトリーには、OSD に関する情報 (所属するクラスター、およびプライベートキーリング) が含まれます。
- **データ**: BlueStore によって直接管理され、すべての OSD データが含まれる残りのデバイスを占有する大容量パーティション。このプライマリーデバイスは、data ディレクトリーのブロックシンボリックリンクで識別されます。

2 つの追加デバイスを使用することもできます。

- **WAL (write-ahead-log) デバイス**: BlueStore 内部ジャーナルまたは write-ahead ログを保存するデバイス。これは、data ディレクトリーの **block.wal** シンボリックリンクによって識別されます。デバイスがプライマリーデバイスよりも高速の場合にのみ WAL デバイスを使用することを検討してください。たとえば、WAL デバイスが SSD ディスクを使用し、プライマリーデバイスが HDD ディスクを使用する場合です。
- **DB デバイス**: BlueStore 内部メタデータを保存するデバイス。組み込み RocksDB データベースは、パフォーマンスを向上させるために、プライマリーデバイスではなく DB デバイスにできるだけ多くのメタデータを配置します。DB デバイスが満杯になると、プライマリーデバイスへのメタデータの追加が開始します。デバイスがプライマリーデバイスよりも高速の場合にのみ DB デバイスを使用することを検討してください。

警告

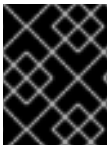
高速デバイスで利用可能なストレージが 1 ギガバイト未満の場合、Red Hat はそれを WAL デバイスとして使用することを推奨します。より高速なデバイスがある場合は、DB デバイスとして使用することを検討してください。BlueStore ジャーナルは常に最速のデバイスに配置されるため、DB デバイスを使用すると、WAL デバイスと同じ利点が得られます。また、追加のメタデータを格納することもできます。

11.3. CEPH BLUESTORE キャッシュ

BlueStore キャッシュババッファーの集合体で、設定によっては OSD デーモンがディスクからの読み込みや書き込みを行う際に、データで埋められることがあります。Red Hat Ceph Storage のデフォルトでは、BlueStore は読み取り時にキャッシュされますが、書き込みは行いません。これは、キャッシュエビクションに関連するオーバーヘッドを回避するために **bluestore_default_buffered_write** オプションが **false** に設定されているためです。

bluestore_default_buffered_write オプションが **true** に設定されていると、データは最初にバッファーに書き込まれ、その後ディスクにコミットされます。その後、書き込みの確認がクライアントに送信されます。これにより、データがエビクトされるまで、キャッシュ内のデータへの読み取り速度が速くなります。

読み取り量の多いワークロードでは、BlueStore キャッシングからすぐに利益を得ることはできません。より多くの読み取りが行われると、キャッシュは時間の経過とともに増大し、後続の読み取りではパフォーマンスが向上するようになります。キャッシュがどのくらいの速さで生成されるかは、BlueStore のブロックおよびデータベースのディスクタイプ、ならびにクライアントのワークロード要件に依存します。



重要

bluestore_default_buffered_write オプションを有効にする前に、Red Hat サポート にお問い合わせください。

キャッシュの age bin

Red Hat Ceph Storage は、さまざまなキャッシュ内の項目を「Age Bin」に関連付け、すべてのキャッシュ項目の想定的な有効期限が表示されます。たとえば、BlueStore onode キャッシュに古い onode エントリーがある場合、単一の大きなオブジェクトに対して読み取りホットワークロードが発生します。その OSD の優先キャッシュは、古い onode エントリーを、ホットオブジェクトのバッファーキャッシュデータよりも低い優先度に分類します。Ceph は通常、特定の優先度で onode を大きく優先する可能性がありますが、このホットワークロードのシナリオでは、バッファーデータのメモリー要求が最初に満たされるように、古い onode にホットワークロードデータよりも優先度の低いレベルが割り当てられる可能性があります。

11.4. CEPH BLUESTORE のサイジングに関する考慮事項

BlueStore OSD を使用して従来のドライブとソリッドステートドライブを混在させる場合には、JusDB 論理ボリューム (**block.db**) のサイズを適切に設定することが重要です。Red Hat では、オブジェクト、ファイル、混合ワークロードで RocksDB の論理ボリュームをブロックサイズの 4% 以下にするこ

とを推奨しています。Red Hat は、JlowsDB および OpenStack のブロックワークロードにおいて、BlueStore ブロックサイズの 1% をサポートしています。たとえば、オブジェクトワークロードのブロックサイズが 1 TB の場合は、最低でも 40 GB の RocksDB 論理ボリュームを作成します。

ドライブタイプを混合しない場合は、個別の RocksDB 論理ボリュームを持つ必要はありません。BlueStore は、RocksDB のサイジングを自動的に管理します。

BlueStore のキャッシュメモリーは、RocksDB、BlueStore のメタデータ、オブジェクトデータのキー/値のペアのメタデータで使用されます。

注記

BlueStore キャッシュのメモリー値は、OSD によってすでに使用されているメモリーフットプリントに追加されます。

11.5. BLUESTORE_MIN_ALLOC_SIZE パラメーターを使用した BLUESTORE の調整

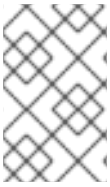
この手順は、新しい OSD または新たにデプロイされた OSD 用です。

BlueStore では、生のパーティションは **bluestore_min_alloc_size** のブロックで割り当て、管理されます。デフォルトでは、**bluestore_min_alloc_size** は **4096** で、HDD および SSD の 4 KiB に相当します。各チャンクの書き込みのない領域は、未加工パーティションに書き込まれる際にゼロで埋められます。これにより、小さいオブジェクトを書き込むなど、ワークロードのサイズが適切に設定されていない場合に未使用領域が無駄になる可能性があります。

この書き込み増幅のペナルティーを回避するために、**bluestore_min_alloc_size** を最小書き込みに一致させることを推奨します。

重要

bluestore_min_alloc_size の値を変更することは推奨しません。サポートが必要な場合は、Red Hat サポート にお問い合わせください。



注記

bluestore_min_alloc_size_ssd 設定および **bluestore_min_alloc_size_hdd** 設定は、それぞれ SSD および HDD に固有のものですが、**bluestore_min_alloc_size** によりその設定が上書きされるため、設定する必要は必要ありません。

前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。
- Ceph モニターおよびマネージャーがクラスターにデプロイされます。
- OSD ノードとして新規にプロビジョニングできるサーバーまたはノード
- Ceph Monitor ノードの管理者キーリング (既存の Ceph OSD ノードを再デプロイする場合)。

手順

1. ブートストラップノードで、**bluestore_min_alloc_size** パラメーターの値を変更します。

   構文

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

例

```
[ceph: root@host01 /]# ceph config set osd.4 bluestore_min_alloc_size_hdd 8192
```

bluestore_min_alloc_size が 8192 バイトに設定されていることを確認できます。これは 8 KiB に相当します。

注記

選択した値は 2 の累乗にする必要があります。

2. OSD のサービスを再起動します。

構文

```
systemctl restart SERVICE_ID
```

例

```
[ceph: root@host01 /]# systemctl restart ceph-499829b4-832f-11eb-8d6d-
001a4a000635@osd.4.service
```

検証

- **ceph daemon** コマンドを使用して設定を確認します。

構文

```
ceph daemon osd.OSD_ID config get bluestore_min_alloc_size__DEVICE_
```

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"
}
```

関連情報

- OSD の削除と追加については、**Red Hat Ceph Storage 運用ガイド** の Ceph Orchestrator を使用した OSD の管理 を参照し、リンクに従ってください。すでにデプロイされている OSD の場合、**bluestore_min_alloc_size** パラメーターを変更して OSD を削除してから再度デプロイする必要があります。

11.6. BLUESTORE 管理ツールを使用して ROCKSDB データベースを再度シャード化する

このリリースでは、BlueStore 管理ツールを使用してデータベースをリシャード化できます。これにより、BlueStore の RocksDB データベースを、OSD を再デプロイせずに、ある形態から別の形態に、さらに複数の列ファミリーに変換します。列ファミリーはデータベース全体と同じ機能がありますが、ユーザーは小規模なデータセットで操作し、異なるオプションを適用することができます。これは、保存されたキーの異なる有効期間を活用します。このキーは、新しいキーを作成したり既存のキーを削除せずに、変換中に移動されます。

OSD を再シャードするには 2 つの方法があります。

1. **rocksdb-resharding.yml** Playbook を使用します。
2. OSD を手動で再シャードします。

前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。
- オブジェクトストア が BlueStore として設定されている。
- OSD ノードがホストにデプロイされている。
- すべてのホストへ root レベルでアクセスできる。
- **ceph-common** パッケージおよび **cephadm** パッケージがすべてのホストにインストールされている。

11.6.1. rocksdb-resharding.yml Playbook を使用する

1. root ユーザーとして、管理ノードで、Playbook がインストールされている **cephadm** フォルダーに移動します。

   例

   ```
   [root@host01 ~]# cd /usr/share/cephadm-ansible
   ```

2. Playbook を実行します。

   構文

   ```
   ansible-playbook -i hosts rocksdb-resharding.yml -e osd_id=OSD_ID -e
   admin_node=HOST_NAME
   ```

   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. 再シャーディングが完了したことを確認します。

   a. 再シャーディングされた OSD を停止します。

      例

      ```
      [ceph: root@host01 /]# ceph orch daemon stop osd.7
      ```

   b. OSD コンテナーを入力します。

      例

      ```
      [root@host03 ~]# cephadm shell --name osd.7
      ```

   c. 再シャーディングを確認します。

      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. OSD を起動します。

例

```
[ceph: root@host01 /]# ceph orch daemon start osd.7
```

11.6.2. OSD を手動で再シャーディングする

1. **cephadm** シェルにログインします。

   例

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

2. 管理ノードから **OSD_ID** とホストの詳細を取得します。

   例

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

3. それぞれのホストに **root** ユーザーでログインし、OSD を停止します。

   構文

   ```
   cephadm unit --name OSD_ID stop
   ```

   例

   ```
   [root@host02 ~]# cephadm unit --name osd.0 stop
   ```

4. 停止中の OSD デーモンコンテナーに入ります。

   構文

   ```
   cephadm shell --name OSD_ID
   ```

   例

   ```
   [root@host02 ~]# cephadm shell --name osd.0
   ```

5. **cephadm shell** にログインし、ファイルシステムの整合性を確認します。

   構文

   ```
   ceph-bluestore-tool --path/var/lib/ceph/osd/ceph-OSD_ID/ fsck
   ```

例

```
[ceph: root@host02 /]# ceph-bluestore-tool --path /var/lib/ceph/osd/ceph-0/ fsck

fsck success
```

6. OSD ノードのシャーディングのステータスを確認します。

構文

```
ceph-bluestore-tool --path /var/lib/ceph/osd/ceph-OSD_ID/ show-sharding
```

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. **ceph-bluestore-tool** コマンドを実行してリシャードします。Red Hat は、コマンドで指定されたパラメーターを使用することを推奨します。

構文

```
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
```

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. OSD ノードのシャーディングのステータスを確認するには、**show-sharding** コマンドを実行します。

構文

```
ceph-bluestore-tool --path /var/lib/ceph/osd/ceph-OSD_ID/ show-sharding
```

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. **cephadm** シェルを終了します。

```
[ceph: root@host02 /]# exit
```

10. それぞれのホストに **root** ユーザーでログインし、OSD を起動します。

構文

```
cephadm unit --name OSD_ID start
```

例

```
[root@host02 ~]# cephadm unit --name osd.0 start
```

関連情報

- 詳細は、Red Hat Ceph Storage インストールガイド を参照してください。

11.7. BLUESTORE 断片化ツール

ストレージ管理者は、BlueStore OSD の断片化レベルを定期的にチェックする必要があります。オフライン OSD またはオンライン OSD の場合は、簡単な 1 つのコマンドを使用して断片化レベルを確認できます。

11.7.1. BlueStore 断片化ツールとは

BlueStore OSD の場合は、基となるストレージデバイスの時間の経過とともに空き領域が断片化されます。一部の断片化は正常ですが、過剰な断片化が生じると、パフォーマンスが低下します。

BlueStore 断片化ツールは、BlueStore OSD の断片化レベルでスコアを生成します。この断片化スコアは 0 から 1 の範囲として指定されます。スコアが 0 の場合は断片化がなく、1 は深刻な断片化を意味します。

表11.1 断片化スコアの意味

スコア	断片化の量
0.0 - 0.4	なしから極小の断片化まで。
0.4 - 0.7	小さく、許容される断片化。
0.7 - 0.9	直感的ですが、安全な断片化です。
0.9 - 1.0	深刻な断片化があり、パフォーマンスの問題が発生することになります。

重要

深刻な断片化があり、問題の解決にサポートが必要な場合は、Red Hat サポート にお問い合わせください。

11.7.2. 断片化の確認

BlueStore OSD の断片化レベルのチェックは、オンラインまたはオフラインで行うことができます。

前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。
- BlueStore OSD

オンラインの BlueStore 断片化スコア

1. 実行中の BlueStore OSD プロセスを検証します。
   a. 簡単なレポート:

      構文

      ceph daemon **OSD_ID** bluestore allocator score block

      例

      [ceph: root@host01 /]# ceph daemon osd.123 bluestore allocator score block

   b. より詳細なレポート:

      構文

      ceph daemon **OSD_ID** bluestore allocator dump block

      例

      [ceph: root@host01 /]# ceph daemon osd.123 bluestore allocator dump block

オフラインの BlueStore 断片化スコア

1. オフラインフラグメンテーションスコアを確認するには、リシャーディングの手順に従います。
   例

[root@host01 ~]# podman exec -it 7fbd6c6293c0 /bin/bash

1. 実行していない BlueStore OSD プロセスを検証します。
   a. 簡単なレポート:

      構文

      ceph-bluestore-tool --path **PATH_TO_OSD_DATA_DIRECTORY** --allocator block free-score

      例

      [root@7fbd6c6293c0 /]# ceph-bluestore-tool --path /var/lib/ceph/osd/ceph-123 --allocator block free-score

   b. より詳細なレポート:

構文

```
ceph-bluestore-tool --path PATH_TO_OSD_DATA_DIRECTORY --allocator block free-dump
block:
{
    "fragmentation_rating": 0.018290238194701977
}
```

例

```
[root@7fbd6c6293c0 /]# 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"
        },
```

関連情報

- 断片化スコアの詳細は、**BlueStore 断片化ツール** を参照してください。
- リシャーディングの詳細は、**BlueStore 管理ツールを使用して RocksDB データベースを再度シャード化する** を参照してください。

11.8. CEPH BLUESTORE BLUEFS

BlueStore ブロックデータベースは、メタデータを RocksDB データベースのキーと値のペアとして格納します。ブロックデータベースは、ストレージデバイス上の小さな BlueFS パーティションに存在します。BlueFS は、RocksDB ファイルを保持するように設計された最小限のファイルシステムです。

BlueFS ファイル

RocksDB が生成する 3 種類のファイルを以下に示します。

- **CURRENT**、**IDENTITY**、**MANIFEST-000011** などの制御ファイル。

- **004112.sst** などの DB テーブルファイル。
- **000038.log** などの先読みログ。

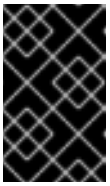
さらに、ディレクトリー構造、ファイルマッピング、および操作ログとして機能する、BlueFS リプレイログ **ino 1** として機能する内部の隠しファイルがあります。

フォールバック階層

BlueFS を使用すると、任意のファイルを任意のデバイスに配置できます。ファイルの一部は、WAL、DB、SLOW などの異なるデバイスに存在する場合もあります。BlueFS がファイルを置く場所には順序があります。ファイルは、1 次ストレージがなくなってから 2 次ストレージに、2 次ストレージがなくなってから 3 次ストレージに配置されます。

特定のファイルの順序は次のとおりです。

- ログ先行書き込み: WAL、DB、SLOW
- リプレイログ **ino 1**: DB、SLOW
- 制御および DB ファイル: DB、SLOW
  - スペース不足時の制御と DB ファイルの順序: SLOW



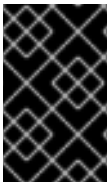
重要

制御および DB ファイルの順序には例外があります。RocksDB は、DB ファイルのスペースが不足していることを検出すると、ファイルを SLOW デバイスに配置するように直接通知します。

11.8.1. bluefs_buffered_io 設定の表示

ストレージ管理者は、**bluefs_buffered_io** パラメーターの現在の設定を表示できます。

Red Hat Ceph Storage では、オプション **bluefs_buffered_io** はデフォルトで **True** に設定されます。このオプションにより、場合によっては BlueFS はバッファー読み取りを実行できるようになり、カーネルページキャッシュが RocksDB ブロック読み取りなどの読み取りのセカンダリーキャッシュとして機能できるようになります。



重要

bluefs_buffered_io の値を変更することは推奨されません。**bluefs_buffered_io** パラメーターを変更する前に、Red Hat サポートアカウントチームにお問い合わせください。

前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。
- Ceph Monitor ノードへの root レベルのアクセス。

手順

1. Cephadm シェルにログインします。

   例

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

2. **bluefs_buffered_io** パラメーターの現在値は、3 つの方法で表示することができます。

方法 1

- 設定データベースに保存されている値を表示します。

  例

  ```
  [ceph: root@host01 /]# ceph config get osd bluefs_buffered_io
  ```

方法 2

- 特定の OSD の設定データベースに保存されている値を表示します。

  構文

  ```
  ceph config get OSD_ID bluefs_buffered_io
  ```

  例

  ```
  [ceph: root@host01 /]# ceph config get osd.2 bluefs_buffered_io
  ```

方法 3

- 実行値が設定データベースに保存されている値と異なる OSD の実行値を表示します。

  構文

  ```
  ceph config show OSD_ID bluefs_buffered_io
  ```

  例

  ```
  [ceph: root@host01 /]# ceph config show osd.3 bluefs_buffered_io
  ```

11.8.2. Ceph OSD の Ceph BlueFS 統計の表示

bluefs stats コマンドを使用して、コロケーションされた Ceph OSD およびコロケーションされていない Ceph OSD に関する BluesFS 関連情報を表示します。

前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。
- オブジェクトストア が BlueStore として設定されている。
- OSD ノードへのルートレベルのアクセス。

手順

1. Cephadm シェルにログインします。

   例

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

2. BlueStore OSD 統計を表示します。

   構文

   ```
   ceph daemon osd.OSD_ID bluefs stats
   ```

   コロケーションされた OSD の例

   ```
   [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
   ```

   コロケーションされていない OSD の例

   ```
   [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
   ```

   ここでは、以下のようになります。

   0: これは専用の WAL デバイス、つまり **block.wal** を参照します。

   1: これは、専用の DB デバイス、つまり **block.db** を指します。

   2: これは、**block** または **slow** であるメインブロックデバイスを指します。

   device size: デバイスの実際のサイズを表します。

   using: 総使用量を表します。これは BlueFS に限定されません。

注記

DB および WAL デバイスは、BlueFS によってのみ使用されます。メインデバイスの場合には、保存された BlueStore データからの使用も含まれます。上記の例では、**2.3 MiB** が BlueStore からのデータです。

wal_total、**db_total**、**slow_total**: これらの値は、上記のデバイスの値を繰り返します。

db_avail: この値は、必要に応じて SLOW デバイスから取得できるバイト数を表します。

使用マトリックス

- **WAL**、**DB**、**SLOW** の行: 特定のファイルが配置される予定だった場所を記述します。
- **LOG** の行: BlueFS リプレイログ **ino 1** について記述します。
- **WAL**、**DB**、**SLOW** の列: データが実際に置かれる場所を記述します。値は割り当て単位です。WAL と DB は、パフォーマンス上の理由上、割り当ての単位が大きくなっています。
- */* の列: **ceph-bluestore-tool** に使用される仮想デバイス **new-db** および **new-wal** に関連します。常に **0 B** と表示されます。
- **REAL** の列: 実際の使用量をバイト単位で示します。
- **FILES** の列: ファイルの数を示します。

MAXIMUMS の列: このテーブルは、使用マトリクスから各エントリーの最大値を取得します。

関連情報

- BlueFS ファイルの詳細は、Ceph BlueStore BlueFS を参照してください。
- BlueStore デバイスの詳細は、Ceph BlueStore デバイス を参照してください。

第12章 CRIMSON （テクノロジープレビュー）

ストレージ管理者として、Crimson プロジェクトは、低遅延、高スループットの永続メモリー、および NVMe テクノロジーという新しい現実に適した **ceph-osd** デーモンの代替を構築する取り組みです。

重要

Crimson 機能はテクノロジープレビューのみの機能です。テクノロジープレビュー機能は、実稼働環境での Red Hat サービスレベルアグリーメント (SLA) ではサポートされておらず、機能的に完全ではない可能性があるため、Red Hat では実稼働環境での使用を推奨していません。テクノロジープレビューの機能は、最新の製品機能をいち早く提供して、開発段階で機能のテストを行いフィードバックを提供していただくことを目的としています。詳細は、Red Hat テクノロジープレビュー機能のサポート範囲 を参照してください。

12.1. CRIMSON の概要

Crimson は、**crimson-osd** のコード名です。これは、マルチコアのスケーラビリティーのための次世代 **ceph-osd** です。DPDK や SPDK などの最先端のテクノロジーを採用し、高速ネットワークおよびストレージデバイスのパフォーマンスを向上させます。BlueStore は引き続き HDD と SSD をサポートします。Crimson は、クラス **ceph-osd** を持つ OSD デーモンの以前のバージョンとの互換性を目指しています。

SeaStar C++ フレームワーク上に構築された Crimson は、コア Ceph オブジェクトストレージデーモン (OSD) コンポーネントの新しい実装であり、**ceph-osd** を置き換えます。**crimson-osd** は、遅延と CPU プロセッサー使用量の増加を最小限に抑えます。高性能の非同期 IO と、相互通信のための操作におけるコンテキストの切り替えとスレッド間通信を最小限に抑えるように設計された新しいスレッドアーキテクチャーを使用します。

注意

Red Hat Ceph Storage 7 の場合、Crimson のみを使用してレプリケートされたプールで RADOS Block Device (RBD) ワークロードをテストできます。本番データには Crimson を使用しないでください。

Crimson の目的

Crimson OSD は、次の目的を備えた OSD デーモンの代替品です。

CPU の過負荷を最小限に抑える

- サイクルまたは IOPS を最小限に抑える。
- コア間通信を最小限に抑える。
- コピーを最小限に抑える。
- カーネルをバイパスし、コンテキストスイッチを回避する。

新規ストレージ技術を有効にする

- ゾーンの名前空間
- 永続メモリー
- 高速 NVMe

SEASTAR の機能

- CPU ごとの単一のリスペクタースレッド
- 非同期 IO
- ユーザー空間で行われるスケジューリング
- ユーザー空間ネットワーキング用の高性能ライブラリーである DPDK の直接サポートが含まれています。

利点

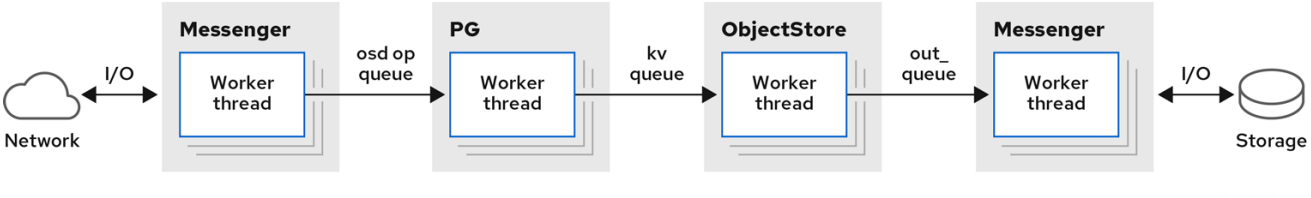
- SeaStore には独立したメタデータコレクションがあります。
- トランザクション
- フラットなオブジェクト名前空間で設定されています。
- オブジェクト名が大きい可能性があります (>1k)。
- 各オブジェクトには、キー > 値のマッピング (文字列 > バイト) とデータペイロードが含まれています。
- COW オブジェクトクローンをサポートします。
- OMAP とオブジェクトの両方の名前空間の順序付きリストをサポートします。

12.2. CRIMSON と CLASSIC CEPH OSD アーキテクチャーの違い

古典的な **ceph-osd** アーキテクチャーでは、メッセンジャースレッドがネットワークからクライアントメッセージを読み取り、メッセージを OP キューに置きます。次に、**osd-op** スレッドプールがメッセージを取得し、トランザクションを作成して、現在のデフォルトの ObjectStore 実装である BlueStore のキューに入れます。次に、BlueStore の **kv_queue** は、このトランザクションとキュー内のその他のものを取得し、**rocksdb** がトランザクションをコミットするのを同時に待ち、完了コールバックをフィニッシャーキューに置きます。次に、フィニッシャースレッドは完了コールバックを取得し、送信するメッセンジャースレッドを置き換えるためにキューに入れます。

これらの各アクションには、キューの内容に対するスレッド間の調整が必要です。**pg state** の場合、競合をロックするために、複数のスレッドが PG の内部メタデータにアクセスする必要がある場合があります。

プロセッサー使用量の増加に伴うこのロック競合は、タスクとコアの数に応じて急速に拡大し、特定のシナリオではすべてのロックポイントがスケーリングのボトルネックになる可能性があります。さらに、これらのロックとキューは、競合していない場合でも遅延コストが発生します。この遅延により、スレッドプールとタスクキューが悪化します。これは、ブックキーピングの作業によりワーカースレッド間でタスクが委任され、ロックによってコンテキストの切り替えが強制される可能性があるためです。

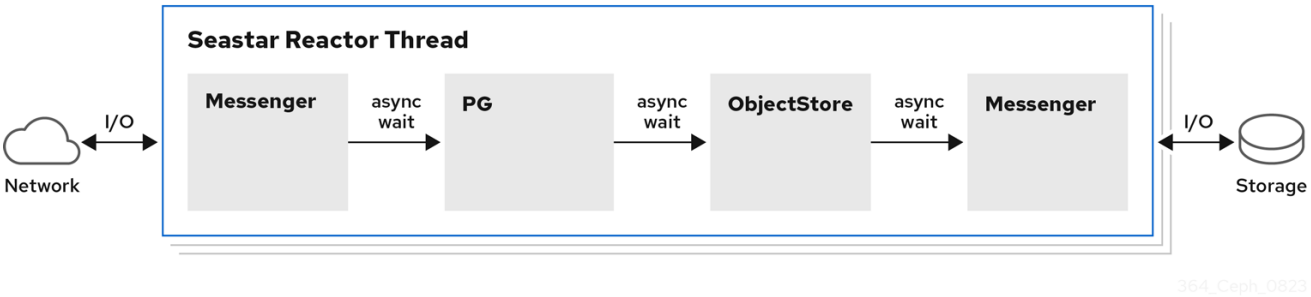


ceph-osd アーキテクチャーとは異なり、Crimson では、コンテキストの切り替えや、基礎となるストレージ操作で要求されない場合はブロックすることなく、単一の I/O 操作を単一のコア上で完了できます。ただし、一部の操作では、非同期プロセスが完了するまで待機できる必要があります。これは、おそらくリカバリーや基盤となるデバイスなどのシステムの状態に応じて非決定的に異なります。

Crimson は Seastar と呼ばれる C++ フレームワークを使用します。これは高度に非同期エンジンであり、通常、各コアに固定された 1 つのスレッドを事前に割り当てます。これらは、コア間で状態を分割し、ロックを回避できるように、コア間で作業を分割します。Seastar では、I/O 操作はターゲットオブジェクトに基づいてスレッドのグループに分割されます。I/O 操作を実行するステージを異なるスレッドグループに分割するのではなく、すべてのパイプラインステージを 1 つのスレッド内で実行します。操作をブロックする必要がある場合、コアの Seastar リアクターは別の同時操作に切り替えて進行します。

理想的には、実行中の各非ブロッキングタスクが完了するか協調的に譲歩するまで CPU を所有するため、すべてのロックとコンテキストスイッチが不要になります。他のスレッドが同時にタスクをプリエンプトすることはできません。データパス内の他のシャードとの通信が必要ない場合、理想的なパフォーマンスは、I/O デバイスが制限に達するまでコアの数に比例して増加します。OSD レベルでは PG がすべての IO をシャーディングするため、この設計は Ceph OSD によく適合します。

ceph-osd とは異なり、**crimson-osd** は、デーモン化オプションが有効になっている場合でも、それ自体をデーモン化しません。サポートされている Linux ディストリビューションはアプリケーションをデーモン化できる **systemd** を使用するため、**crimson-osd** をデーモン化しないでください。**sysvinit** では、**start-stop-daemon** を使用して **crimson-osd** をデーモン化します。



ObjectStore バックエンド

crimson-osd は、ネイティブオブジェクトストアバックエンドとエイリアン化されたオブジェクトストアバックエンドの両方を提供します。ネイティブオブジェクトストアバックエンドは、Seastar リアクターで I/O を実行します。

Crimson では次の 3 つの ObjectStore バックエンドがサポートされています。

- **AlienStore**: オブジェクトストアの以前のバージョン、つまり BlueStore との互換性を提供します。
- **CyanStore**: 揮発性メモリーによって実装されるテスト用のダミーバックエンド。このオブジェクトストアは、クラシック OSD の **memstore** をモデルにしています。
- **SeaStore**: Crimson OSD 用に特別に設計された新しいオブジェクトストア。複数のシャードのサポートに向けたパスは、バックエンドの特定の目標によって異なります。

以下は、他の 2 つの古典的な OSD ObjectStore バックエンドです。

- **MemStore**: バックエンドオブジェクトストアとしてのメモリー。
- **BlueStore**: クラシック **ceph-osd** で使用されるオブジェクトストア。

12.3. CRIMSON メトリクス

Crimson には、統計とメトリクスをレポートする 3 つの方法があります。

- マネージャーに報告された PG 統計。
- Prometheus テキストプロトコル。
- **asock** コマンド。

マネージャーに報告された PG 統計

Crimson は、**MPGStats** メッセージ内の **per-pg**、**per-pool**、および **per-osd** 統計を収集して、Ceph Manager に送信します。

Prometheus テキストプロトコル

--prometheus-port コマンドラインオプションを使用して、リスニングポートとアドレスを設定します。

asock コマンド

メトリクスをダンプするための管理ソケットコマンドが提供されています。

構文

```
ceph tell OSD_ID dump_metrics
ceph tell OSD_ID dump_metrics reactor_utilization
```

例

```
[ceph: root@host01 /]# ceph tell osd.0 dump_metrics
[ceph: root@host01 /]# ceph tell osd.0 dump_metrics reactor_utilization
```

ここで、**reactor_utilization** は、ダンプされたメトリクスを接頭辞でフィルタリングするためのオプションの文字列です。

12.4. CRIMSON 設定オプション

Seastar 固有のコマンドラインオプションは **crimson-osd --help-seastar** コマンドを実行します。Crimson の設定に使用できるオプションは次のとおりです。

--crimson, 説明

ceph-osd の代わりに **crimson-osd** を起動します。

--nodaemon, 説明

サービスをデーモン化しません。

--redirect-output, 説明

stdout および **stderr** を **out/$type.$num.stdout** にリダイレクトします。

--osd-args, 説明

crimson-osd または **ceph-osd** に追加のコマンドラインオプションを渡します。このオプションは、Seastar オプションを **crimson-osd** に渡す場合に便利です。たとえば、**--osd-args "--memory 2G"** を指定して、使用するメモリーの量を設定できます。

--cyanstore, 説明

オブジェクトストアバックエンドとして CyanStore を使用します。

--bluestore, 説明

エイリアン化された BlueStore をオブジェクトストアバックエンドとして使用します。**--bluestore** は、デフォルトのメモリーストアです。

--memstore, 説明

エイリアン化された MemStore をオブジェクトストアバックエンドとして使用します。

--seastore, 説明

バックエンドオブジェクトストアとして SeaStore を使用します。

--seastore-devs, 説明

SeaStore によって使用されるブロックデバイスを指定します。

--seastore-secondary-devs, 説明

オプションです。SeaStore は複数のデバイスをサポートしています。このオプションにブロックデバイスを渡すことで、この機能を有効にします。

--seastore-secondary-devs-type, 説明

オプションです。セカンダリーデバイスのタイプを指定します。**--seastore-devs** に渡されたセカンダリーデバイスがメインデバイスよりも遅い場合は、より高速なデバイスのコールドデータは時間の経過とともにより遅いデバイスに追い出されます。有効なタイプには、**HDD**、**SSD**、**(default)**、**ZNS**、および **RANDOM_BLOCK_SSD** があります。セカンダリーデバイスはメインデバイスよりも高速であってはいけないことに注意してください。

12.5. CRIMSON の設定

新規ストレージクラスターをインストールして、**crimson-osd** を設定します。bootstrap オプションを使用して、新しいクラスターをインストールします。このクラスターは実験段階にあるため、アップグレードできません。警告: データが失われる可能性があるため、実稼働データは使用しないでください。

前提条件

- 最初の Ceph Monitor コンテナーの IP アドレス。これはストレージクラスターの最初のノードの IP アドレスでもあります。
- **registry.redhat.io** へのログインアクセス。
- 少なくとも 10 GB の空き容量がある **/var/lib/containers/**。
- 全ノードへの root レベルのアクセス。

手順

1. ブートストラップ中に **--image** フラグを使用して Crimson ビルドを使用します。

   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. **cephadm** シェルにログインします。

   例

   ```
   [root@host 01 ~]# cephadm shell
   ```

3. Crimson を実験的機能としてグローバルに有効にします。

   例

   ```
   [ceph: root@host01 /]# ceph config set global
   'enable_experimental_unrecoverable_data_corrupting_features' crimson
   ```

   この手順は、**crimson** を有効にします。Crimson は非常に実験的であり、クラッシュやデータ損失などの誤動作が予想されます。

4. OSD マップフラグを有効にします。

   例

   ```
   [ceph: root@host01 /]# ceph osd set-allow-crimson --yes-i-really-mean-it
   ```

   モニターにより、**crimson-osd** は **--yes-i-really-mean-it** フラグでのみ起動できます。

5. モニターの Crimson パラメーターを有効にして、デフォルトのプールが Crimson プールとして作成されるように指示します。

   例

   ```
   [ceph: root@host01 /]#  ceph config set mon osd_pool_default_crimson true
   ```

   crimson-osd は、非 crimson プールの配置グループ (PG) を開始しません。

12.6. CRIMSON 設定パラメーター

以下は、Crimson の設定に使用できるパラメーターです。

crimson_osd_obc_lru_size

説明

キャッシュする obc の数。

型

uint

デフォルト

10

crimson_osd_scheduler_concurrency

説明

同時 IO 操作の最大数。無制限の場合は **0**。

型

uint

デフォルト

0

crimson_alien_op_num_threads

説明

エイリアン化された ObjectStore を提供するためのスレッドの数。

型

uint

デフォルト

6

crimson_seastar_smp

説明

OSD に使用する Seastar Reactor スレッドの数。

型

uint

デフォルト

1

crimson_alien_thread_cpu_cores

説明

文字列 cpuset(7) 形式で alienstore スレッドが実行される CPU コア。

型

String

seastore_segment_size

説明

Segment Manager に使用するセグメントサイズ。

型

サイズ

デフォルト

64_M

seastore_device_size

説明

SegmentManager ブロックファイルが作成された場合に使用する合計サイズ。

型

サイズ

デフォルト

50_G

seastore_block_create

説明

SegmentManager ファイルが存在しない場合は作成します。

型

Boolean

デフォルト

true

seastore_journal_batch_capacity

説明

仕訳バッチ内のレコードの数制限。

型

uint

デフォルト

16

seastore_journal_batch_flush_size

説明

ジャーナルバッチを強制的にフラッシュするサイズのしきい値。

型

サイズ

デフォルト

16_M

seastore_journal_iodepth_limit

説明

ジャーナルレコードを送信するための IO 深さの制限。

型

uint

デフォルト

5

seastore_journal_batch_preferred_fullness

説明

ジャーナルバッチをフラッシュするためのレコード満杯のしきい値。

型

浮動小数点 (Float)

デフォルト

0.95

seastore_default_max_object_size

説明

seastore オブジェクトのデータのデフォルトの論理アドレススペース予約。

型

uint

デフォルト

16777216

seastore_default_object_metadata_reservation

説明

seastore オブジェクトのメタデータのデフォルトの論理アドレススペース予約。

型

uint

デフォルト

16777216

seastore_cache_lru_size

説明

キャッシュに保持するエクステントのサイズ (バイト単位)。

型

サイズ

デフォルト

64_M

seastore_cbjournal_size

説明

CircularBoundedJournal が作成された場合に使用する合計サイズ。seastore_main_device_type が RANDOM_BLOCK の場合にのみ有効です。

型

サイズ

デフォルト

5_G

seastore_obj_data_write_amplification

説明

書き込みサイズに対する合計エクステントサイズの比率がこの値を超える場合、エクステントを分割します。

型

浮動小数点 (Float)

デフォルト

1.25

seastore_max_concurrent_transactions

説明

seastore が許可する最大同時トランザクション。

型

uint

デフォルト

8

seastore_main_device_type

説明

seastore が使用する主なデバイスタイプ (SSD または RANDOM_BLOCK_SSD)。

型

String

デフォルト

SSD

seastore_multiple_tiers_stop_evict_ratio

説明

メイン層の使用率がこの値より小さい場合、コールド層へのコールドデータの削除を停止します。

型

浮動小数点 (Float)

デフォルト

0.5

seastore_multiple_tiers_default_evict_ratio

説明

メイン層の使用率がこの値に達すると、コールドデータのコールド層への削除が開始されます。

型

浮動小数点 (Float)

デフォルト

0.6

seastore_multiple_tiers_fast_evict_ratio

説明

メイン層の使用比率がこの値に達すると、高速エビクションを開始します。

型

浮動小数点 (Float)

デフォルト

0.7

12.7. PROFILING CRIMSON

Profiling Crimson は、Crimson でのパフォーマンステストを行う手法です。次の 2 種類のプロファイリングがサポートされています。

- フレキシブル I/O (FIO): **crimson-store-nbd** は、FIO で使用する NBD サーバーとして、設定可能な **FuturizedStore** 内部を示します。
- Ceph ベンチマークツール (CBT): Ceph クラスターのパフォーマンスをテストするための Python のテストハーネス。

手順

1. **libnbd** をインストールし、FIO をコンパイルします。

   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. **crimson-store-nbd** をビルドします。

   例

   ```
   [root@host01 ~]# cd build
   [root@host01 ~]# ninja crimson-store-nbd
   ```

3. ブロックデバイスを使用して **crimson-store-nbd** サーバーを実行します。**/dev/nvme1n1** のような RAW デバイスへのパスを指定します。

   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. nbd.fio という名前の FIO ジョブを作成します。

   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. コンパイルされた FIO を使用して Crimson オブジェクトをテストします。

   例

```
[root@host01 ~]# ./fio nbd.fio
```

Ceph Benchmarking Tool (CBT)

2 つのブランチに対して同じテストを実行します。1 つは **main** (マスター)、もう 1 つは選択した **topic** ブランチです。テスト結果を比較します。すべてのテストケースとともに、2 セットのテスト結果を比較するときに回帰を実行する必要があるかどうかをチェックする一連のルールが定義されます。回帰の可能性が見つかった場合は、ルールと対応するテスト結果が強調表示されます。

手順

1. メインブランチとトピックブランチから、**make crimson osd** を実行します。

   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. テスト結果を比較します。

   例

```
[root@host01 ~]# ~/dev/cbt/compare.py -b /tmp/baseline -a /tmp/yap -v
```

第13章 CEPHADM のトラブルシューティング

ストレージ管理者は、Red Hat Ceph Storage クラスターのトラブルシューティングを行うことができます。場合によっては、Cephadm コマンドが失敗した理由や、特定のサービスが適切に実行されない理由を調査する必要があります。

13.1. CEPHADM の一時停止または無効化

Cephadm が期待どおりに動作しない場合は、次のコマンドを使用して、ほとんどのバックグラウンドアクティビティーを一時停止できます。

例

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

これにより、変更はすべて停止しますが、Cephadm は定期的にホストをチェックして、デーモンとデバイスのインベントリーを更新します。

Cephadm を完全に無効にする場合は、次のコマンドを実行します。

例

```
[ceph: root@host01 /]# ceph orch set backend ''
[ceph: root@host01 /]# ceph mgr module disable cephadm
```

以前にデプロイされたデーモンコンテナーは引き続き存在し、以前と同じように起動することに注意してください。

クラスター内で Cephadm を再度有効にするには、次のコマンドを実行します。

例

```
[ceph: root@host01 /]# ceph mgr module enable cephadm
[ceph: root@host01 /]# ceph orch set backend cephadm
```

13.2. サービスごとおよびデーモンごとのイベント

Cephadm は、失敗したデーモンのデプロイのデバッグを支援するために、サービスごとおよびデーモンごとにイベントを保存します。これらのイベントには、関連する情報が含まれていることがよくあります。

サービスごと

構文

```
ceph orch ls --service_name SERVICE_NAME --format yaml
```

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"'
```

デーモンごと

構文

```
ceph orch ps --service-name SERVICE_NAME --daemon-id DAEMON_ID --format yaml
```

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. CEPHADM ログの確認

次のコマンドを使用して、Cephadm のログをリアルタイムで監視できます。

例

```
[ceph: root@host01 /]# ceph -W cephadm
```

次のコマンドを使用すると、最後のいくつかのメッセージを確認できます。

例

```
[ceph: root@host01 /]# ceph log last cephadm
```

ファイルへのロギングを有効にしている場合、モニターホストに **ceph.cephadm.log** という Cephadm ログファイルが表示されます。

13.4. ログファイルの収集

journalctl コマンドを使用して、すべてのデーモンのログファイルを収集できます。

注記

これらのコマンドはすべて、**cephadm** シェルの外部で実行する必要があります。

注記

デフォルトでは、Cephadm はログを journald に格納します。つまり、デーモンログは **/var/log/ceph** では利用できなくなります。

- 特定のデーモンのログファイルを読み取るには、次のコマンドを実行します。

  構文

  ```
  cephadm logs --name DAEMON_NAME
  ```

  例

  ```
  [root@host01 ~]# cephadm logs --name cephfs.hostname.ppdhsz
  ```

注記

このコマンドは、デーモンが実行されているホストと同じホスト上で実行すると機能します。

- 別のホストで実行されている特定のデーモンのログファイルを読み取るには、次のコマンドを実行します。

  構文

  ```
  cephadm logs --fsid FSID --name DAEMON_NAME
  ```

  例

  ```
  [root@host01 ~]# cephadm logs --fsid 2d2fd136-6df1-11ea-ae74-002590e526e8 --name
  cephfs.hostname.ppdhsz
  ```

  ここで、**fsid** は **ceph status** コマンドによって提供されるクラスター ID です。

- 特定のホスト上のすべてのデーモンのすべてのログファイルをフェッチするには、次のコマンドを実行します。

  構文

  ```
  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
  ```

  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. SYSTEMD ステータスの収集

- systemd ユニットの状態を出力するには、次のコマンドを実行します。

  例

```
[root@host01 ~]$ systemctl status ceph-a538d494-fb2a-48e4-82c8-
b91c37bb0684@mon.host01.service
```

13.6. ダウンロードされたすべてのコンテナーイメージのリスト表示

ホストにダウンロードされたすべてのコンテナーイメージをリスト表示するには、次のコマンドを実行します。

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. コンテナーの手動による実行

Cephadm はコンテナーを実行する小さなラッパーを作成します。コンテナー実行コマンドを実行するには、**/var/lib/ceph/CLUSTER_FSID/SERVICE_NAME/unit** を参照してください。

SSH エラーの分析

次のエラーが表示された場合:

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
```

次のオプションを試して、問題のトラブルシューティングを行います。

- Cephadm に SSH アイデンティティーキーがあることを確認するには、次のコマンドを実行します。

  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
```

上記のコマンドが失敗した場合、Cephadm にはキーがありません。SSH キーを生成するには、次のコマンドを実行します。

例

```
[ceph: root@host01 /]# chmod 0600 ~/cephadm_private_key
```

または、以下を実行します。

例

```
[ceph: root@host01 /]# cat ~/cephadm_private_key | ceph cephadm set-ssk-key -i-
```

- SSH 設定が正しいことを確認するには、次のコマンドを実行します。

  例

  ```
  [ceph: root@host01 /]# ceph cephadm get-ssh-config
  ```

- ホストへの接続を確認するには、次のコマンドを実行します。

  例

  ```
  [ceph: root@host01 /]# ssh -F config -i ~/cephadm_private_key root@host01
  ```

公開鍵が authorized_keys にあることを確認します。

公開鍵が **authorized_keys** ファイルにあることを確認するには、次のコマンドを実行します。

例

```
[ceph: root@host01 /]# ceph cephadm get-pub-key
[ceph: root@host01 /]# grep "`cat ~/ceph.pub`" /root/.ssh/authorized_keys
```

13.8. CIDR ネットワークエラー

スーパーネット化とも呼ばれる Classless inter domain routing (CIDR) は、Internet Protocol (IP) アドレスを割り当てる方法です。Cephadm ログエントリーは、アドレス配布の効率を向上させ、クラス A、クラス B、およびクラス C のネットワークに基づく以前のシステムを置き換える現在の状態を示します。次のエラーのいずれかが表示された場合:

ERROR: Failed to infer CIDR network for mon ip *; pass --skip-mon-network to configure it later

または、以下を実行します。

Must set public_network config option or specify a CIDR network, ceph addrvec, or plain IP

次のコマンドを実行する必要があります。

例

```
[ceph: root@host01 /]# ceph config set host public_network hostnetwork
```

13.9. 管理ソケットへのアクセス

各 Ceph デーモンは MON をバイパスする管理ソケットを提供します。

管理ソケットにアクセスするには、ホストのデーモンコンテナーにアクセスします。

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. MGR デーモンの手動によるデプロイ

Cephadm は Red Hat Ceph Storage クラスターを管理するために **mgr** デーモンを必要とします。Red Hat Ceph Storage クラスターの最後の **mgr** デーモンが削除された場合は、Red Hat Ceph Storage クラスターのランダムホストに **mgr** デーモンを手動でデプロイできます。

前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。
- すべてのノードへの root レベルのアクセス。
- ホストがクラスターに追加されている。

手順

1. Cephadm シェルにログインします。

   例

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

2. 次のコマンドを使用して、Cephadm が新しい MGR デーモンを削除しないように、Cephadm スケジューラーを無効にします。

   例

   ```
   [ceph: root@host01 /]# ceph config-key set mgr/cephadm/pause true
   ```

3. 新しい MGR デーモンの **auth** エントリーを取得または作成します。

   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. **ceph.conf** ファイルを開きます。

   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. コンテナーイメージを取得します。

   例

```
[ceph: root@host01 /]# ceph config get "mgr.host01.smfvfd1" container_image
```

6. **config-json.json** ファイルを作成し、以下を追加します。

   注記

   ceph config generate-minimal-conf コマンドの出力の値を使用します。

   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. Cephadm シェルを終了します。

   例

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

8. MGR デーモンをデプロイします。

   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
```

検証

Cephadm シェルで、次のコマンドを実行します。

例

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

新しい **mgr** デーモンが追加されたことがわかります。

第14章 CEPHADM の操作

ストレージ管理者は、Red Hat Ceph Storage クラスターで Cephadm 操作を実行できます。

14.1. CEPHADM ログメッセージの監視

Cephadm は cephadm クラスターのログチャネルにログを記録するので、リアルタイムで進捗を監視できます。

- 進行状況をリアルタイムで監視するには、次のコマンドを実行します。

  例

  ```
  [ceph: root@host01 /]# ceph -W cephadm
  ```

  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
  ```

- デフォルトでは、ログには情報レベル以上のイベントが表示されます。デバッグレベルのメッセージを表示するには、次のコマンドを実行します。

  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
  ```

- デバッグレベルをデフォルトの **info** に戻します。

  例

  ```
  [ceph: root@host01 /]# ceph config set mgr mgr/cephadm/log_to_cluster_level info
  ```

- 最近のイベントを表示するには、次のコマンドを実行します。

  例

  ```
  [ceph: root@host01 /]# ceph log last cephadm
  ```

これらのイベントは、モニターホスト上の **ceph.cephadm.log** ファイルおよびモニターデーモンの **stderr** にも記録されます。

14.2. CEPH デーモンログ

stderr またはファイルを介して Ceph デーモンログを表示できます。

stdout へのロギング

従来、Ceph デーモンは **/var/log/ceph** にログを記録していました。デフォルトでは、Cephadm デーモンは **stderr** にログを記録し、ログはコンテナーランタイム環境によってキャプチャーされます。ほとんどのシステムでは、デフォルトでは、これらのログは **journald** に送信され、**journalctl** コマンドを使用してアクセスできます。

- たとえば、ID **5c5a50ae-272a-455d-99e9-32c6a013e694** のストレージクラスターの **host01** 上のデーモンのログを表示するには、次のようにします。

  例

  ```
  [ceph: root@host01 /]# journalctl -u ceph-5c5a50ae-272a-455d-99e9-
  32c6a013e694@host01
  ```

これは、ロギングレベルが低い場合に、通常の Cephadm 操作で適切に機能します。

- **stderr** へのロギングを無効にするには、次の値を設定します。

  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
  ```

ファイルへのロギング

また、**stderr** ではなくファイルにログを記録するように Ceph デーモンを設定することもできます。ファイルにロギングする場合、Ceph ログは **/var/log/ceph/CLUSTER_FSID** にあります。

- ファイルへのロギングを有効にするには、次の値を設定します。

  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
  ```

注記

Red Hat では、二重ログを回避するために **stderr** へのロギングを無効にすることを推奨します。

重要

現在、デフォルト以外のパスへのログローテーションはサポートされていません。

デフォルトでは、Cephadm は各ホストでログローテーションを設定し、これらのファイルをローテーションします。**/etc/logrotate.d/ceph.CLUSTER_FSID** を変更することで、ロギングの保持スケジュールを設定できます。

14.3. データの場所

Cephadm デーモンのデータとログは、古いバージョンの Ceph とは少し異なる場所にあります。

- **/var/log/ceph/CLUSTER_FSID** には、すべてのストレージクラスターログが含まれます。デフォルトでは、Cephadm は **stderr** とコンテナーランタイムを介してログを記録するため、これらのログは通常存在しません。
- **/var/lib/ceph/CLUSTER_FSID** には、ログ以外のすべてのクラスターデーモンのデータが含まれます。
- **var/lib/ceph/CLUSTER_FSID/DAEMON_NAME** には、特定のデーモンのすべてのデータが含まれています。
- **/var/lib/ceph/CLUSTER_FSID/crash** には、ストレージクラスターのクラッシュレポートが含まれます。
- **/var/lib/ceph/CLUSTER_FSID/removed** には、ステートフルデーモンの古いデーモンのデータディレクトリーが含まれています (Cephadm によって削除されたモニターや Prometheus など)。

ディスク使用量

いくつかの Ceph デーモンは、**/var/lib/ceph** に大量のデータを格納することがあります (特にモニターと Prometheus デーモン)。したがって、Red Hat は、ルートファイルシステムがいっぱいにならないように、このディレクトリーを独自のディスク、パーティション、または論理ボリュームに移動することを推奨します。

14.4. CEPHADM カスタム設定ファイル

Cephadm では、デーモン用のさまざまな設定ファイルを指定できます。設定ファイルの内容と、それをマウントするデーモンのコンテナー内の場所の両方を指定する必要があります。

YAML 仕様は、指定されたカスタム設定ファイルを使用して適用されます。Cephadm は、設定ファイルが指定されているデーモンを再デプロイします。次に、これらのファイルは、デーモンのコンテナー内の指定された場所にマウントされます。

- カスタム設定ファイルを使用して YAML 仕様を適用できます。

  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
ZWQgZGlhbSB2b2x1cHR1YS4gQXQgdmVyby4gQXQgYWNjdXNhbSBldCBqdXN0byBkdW8=
-----END PRIVATE KEY-----
-----BEGIN CERTIFICATE-----
V2VyIGRhcyBsaWVzdCBpc3QgZG9vZi4gTG9yZW0gaXBzdW0gZG9sb3Igc2l0IGFtZXQsIGNv
bnNldGV0dXIgc2FkaXBzY2luZyBlbGl0ciwgc2VkIGRpYW0gbm9udW15IGVpcm1vZCB0ZW1w
b3IgaW52aWR1bnQgdXQgbGFib3JlIGV0IGRvbG9yZSBtYWduYSBhbGlxdXlhbSBlcmF0LCBz
ZWQgZGlhbSB2b2x1cHR1YS4gQXQgdmVyby4gQXQgYWNjdXNhbSBldCBqdXN0byBkdW8=
-----END CERTIFICATE-----
```

- デーモンのコンテナー内に新しい設定ファイルをマウントできます。

  構文

```
ceph orch redeploy SERVICE_NAME
```

  例

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

第15章 CEPHADM ヘルスチェック

ストレージ管理者は、Cephadm モジュールによって提供される追加のヘルスチェックを使用して Red Hat Ceph Storage クラスターを監視できます。これは、ストレージクラスターによって提供されるデフォルトのヘルスチェックの補足です。

15.1. CEPHADM 操作のヘルスチェック

ヘルスチェックは、Cephadm モジュールがアクティブなときに実行されます。次のヘルス警告を受け取る場合があります。

CEPHADM_PAUSED

Cephadm のバックグラウンド作業は、**ceph orch pause** コマンドで一時停止します。Cephadm は、ホストとデーモンの状態を確認するなどのパッシブ監視アクティビティーを実行し続けますが、デーモンのデプロイや削除などの変更は行いません。**ceph orch resume** コマンドを使用して、Cephadm の作業を再開できます。

CEPHADM_STRAY_HOST

1 つ以上のホストが Ceph デーモンを実行していますが、Cephadm モジュールによって管理されるホストとして登録されていません。これは、これらのサービスが現在 Cephadm によって管理されていないことを意味します。たとえば、**ceph orch ps** コマンドに含まれる再起動とアップグレードなどです。**ceph orch host add HOST_NAME** コマンドを使用してホストを管理できますが、リモートホストへの SSH アクセスが設定されていることを確認してください。または、手動でホストに接続し、そのホスト上のサービスが削除または Cephadm によって管理されているホストに移行されるようにすることもできます。この警告は、設定 **ceph config set mgr mgr/cephadm/warn_on_stray_hosts false** で無効にすることもできます。

CEPHADM_STRAY_DAEMON

1 つ以上の Ceph デーモンが動作中ですが、Cephadm モジュールによって管理されていません。これは、別のツールを使用してデプロイされたか、手動で開始されたためです。これらのサービスは、現在 Cephadm によって管理されていません。たとえば、**ceph orch ps** コマンドに含まれる再起動とアップグレードなどです。

デーモンがモニターまたは OSD デーモンであるステートフルなデーモンである場合、これらのデーモンは Cephadm によって採用される必要があります。ステートレスデーモンの場合は、**ceph orch apply** コマンドで新しいデーモンをプロビジョニングし、アンマネージデーモンを停止できます。

このヘルス警告は、設定 **ceph config set mgr mgr/cephadm/warn_on_stray_daemons false** で無効にすることができます。

CEPHADM_HOST_CHECK_FAILED

1 つ以上のホストが基本的な Cephadm ホストチェックに失敗しています。name: value を検証します

- ホストは到達可能で、Cephadm を実行することができます。
- ホストは、Podman であるコンテナーランタイムの機能、時間同期の機能など、基本的な前提条件を満たしています。このテストが失敗した場合、Cephadm はそのホスト上のサービスを管理できません。

このチェックは、**ceph cephadm check-host HOST_NAME** コマンドで手動で実行できます。壊れたホストを管理から削除するには、**ceph orch host rm HOST_NAME** コマンドを使用します。このヘルス警告は、設定 **ceph config set mgr mgr/cephadm/warn_on_failed_host_check false** で無効にすることができます。

15.2. CEPHADM 設定のヘルスチェック

Cephadm は、OS、ディスク、および NIC の状態を把握するために、ストレージクラスター内の各ホストを定期的にスキャンします。これらの事実は、ストレージクラスター内のホスト全体の整合性について分析され、設定の異常を特定します。設定のチェックはオプション機能です。

- この機能は、次のコマンドで有効にできます。

  例

  ```
  [ceph: root@host01 /]# ceph config set mgr mgr/cephadm/config_checks_enabled true
  ```

設定チェックは、各ホストスキャンの後にトリガーされます。このスキャンは 1 分間です。

- **ceph -W cephadm** コマンドは、現在の状態のログエントリーと設定チェックの結果を次のように表示します。
  無効な状態

  例

  ```
  ALL cephadm checks are disabled, use 'ceph config set mgr
  mgr/cephadm/config_checks_enabled true' to enable
  ```

  有効な状態

  例

  ```
  CEPHADM 8/8 checks enabled and executed (0 bypassed, 0 disabled). No issues detected
  ```

  設定チェック自体は、いくつかの **cephadm** サブコマンドによって管理されます。

- 設定のチェックが有効になっているかどうかを確認するには、次のコマンドを実行します。

  例

  ```
  [ceph: root@host01 /]# ceph cephadm config-check status
  ```

  このコマンドは、設定チェッカーのステータスを **Enabled** または **Disabled** のいずれかとして返します。

- すべての設定チェックとその現在の状態をリスト表示するには、次のコマンドを実行します。

  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
```

各設定チェックは、次のように記述されます。

CEPHADM_CHECK_KERNEL_LSM

ストレージクラスター内の各ホストは、同じ Linux セキュリティーモジュール (LSM) の状態で動作すると予想されます。たとえば、大半のホストが **enforcing** モードの SELINUX で実行されている場合、このモードで実行されていないホストには異常フラグが付けられ、警告状態のヘルスチェックが発生します。

CEPHADM_CHECK_SUBSCRIPTION

このチェックは、ベンダーサブスクリプションのステータスに関連します。このチェックは、Red Hat Enterprise Linux を使用するホストに対してのみ実行されますが、パッチと更新が利用可能になるように、すべてのホストがアクティブなサブスクリプションの対象になっていることを確認するのに役立ちます。

CEPHADM_CHECK_PUBLIC_MEMBERSHIP

クラスターのすべてのメンバーは、少なくとも 1 つのパブリックネットワークサブネットに NIC を設定している必要があります。パブリックネットワーク上にないホストは、パフォーマンスに影響する可能性のあるルーティングに依存します。

CEPHADM_CHECK_MTU

OSD 上の NIC の最大伝送ユニット (MTU) は、一貫したパフォーマンスの重要な要素となります。このチェックでは、OSD サービスを実行しているホストを調べて、MTU がクラスター内で一貫して設定されていることを確認します。これは、大多数のホストが使用している MTU 設定を確立することによって決定し、異常があれば Ceph ヘルスチェックを行います。

CEPHADM_CHECK_LINKSPEED

MTU チェックと同様に、リンクスピードの整合性も、一貫したクラスターパフォーマンスの要因になります。このチェックは、OSD ホストの大部分で共有されるリンク速度を決定し、より低いリンク速度で設定されているホストのヘルスチェックを行います。

CEPHADM_CHECK_NETWORK_MISSING

public_network および **cluster_network** 設定は、IPv4 および IPv6 のサブネット定義をサポートします。これらの設定がストレージクラスター内のどのホストにも見つからない場合は、ヘルスチェックが発生します。

CEPHADM_CHECK_CEPH_RELEASE

通常の操作では、Ceph クラスターは同じ Ceph リリースでデーモンを実行する必要があります (例: すべて Red Hat Ceph Storage クラスター 5 リリース)。このチェックは、各デーモンのアクティブなリリースを調べ、異常をヘルスチェックとして報告します。クラスター内でアップグレードプロセスがアクティブな場合、このチェックは省略されます。

CEPHADM_CHECK_KERNEL_VERSION

OS カーネルのバージョンの整合性が、全ホストでチェックされます。これまでと同様に、大多数のホストを異常特定のベースとして使用されます。

第16章 CEPHADM-ANSIBLE モジュールを使用した RED HAT CEPH STORAGE クラスターの管理

ストレージ管理者として、Ansible Playbook で **cephadm-ansible** モジュールを使用して、Red Hat Ceph Storage クラスターを管理することができます。**cephadm-ansible** パッケージは、クラスターを管理するための独自の Ansible Playbook を作成できるように、**cephadm** 呼び出しをラップするいくつかのモジュールを提供します。

注記

現時点では、**cephadm-ansible** モジュールは最も重要なタスクのみをサポートしています。**cephadm-ansible** モジュールでカバーされていない操作は、Playbook で **command** または **shell** Ansible モジュールを使用して完了する必要があります。

16.1. CEPHADM-ANSIBLE モジュール

cephadm-ansible モジュールは、**cephadm** および **ceph orch** コマンドのラッパーを提供することで、Ansible Playbook の作成を簡素化するモジュールのコレクションです。モジュールを使用して独自の Ansible Playbook を作成し、1 つ以上のモジュールを使用してクラスターを管理できます。

cephadm-ansible パッケージには、次のモジュールが含まれています。

- **cephadm_bootstrap**
- **ceph_orch_host**
- **ceph_config**
- **ceph_orch_apply**
- **ceph_orch_daemon**
- **cephadm_registry_login**

16.2. CEPHADM-ANSIBLE モジュールのオプション

次の表に、**cephadm-ansible** モジュールで使用可能なオプションを示します。Ansible Playbook でモジュールを使用する場合は、必須としてリストされているオプションを設定する必要があります。デフォルト値 **true** でリストされているオプションは、モジュールの使用時にオプションが自動的に設定され、Playbook で指定する必要がないことを示します。たとえば、**cephadm_bootstrap** モジュールの場合、**dashboard: false** を設定しない限り、Ceph Dashboard がインストールされます。

表16.1 cephadm_bootstrap モジュールで利用可能なオプション

cephadm_bootstrap	説明	必須	デフォルト
mon_ip	Ceph Monitor の IP アドレス。	true	
image	Ceph コンテナーイメージ。	false	

cephadm_bootstrap	説明	必須	デフォルト
docker	**podman** の代わりに **docker** を使用します。	false	
fsid	Ceph FSID を定義します。	false	
pull	Ceph コンテナーイメージをプルします。	false	true
dashboard	Ceph Dashboard をデプロイします。	false	true
dashboard_user	特定の Ceph Dashboard ユーザーを指定します。	false	
dashboard_password	Ceph Dashboard のパスワード。	false	
monitoring	モニタリングスタックをデプロイします。	false	true
firewalld	firewalld を使用してファイアウォールルールを管理します。	false	true
allow_overwrite	既存の --output-config、--output-keyring、または --output-pub-ssh-key ファイルの上書きを許可します。	false	false
registry_url	カスタムレジストリーの URL。	false	
registry_username	カスタムレジストリーのユーザー名。	false	
registry_password	カスタムレジストリーのパスワード。	false	
registry_json	カスタムレジストリーログイン情報を含む JSON ファイル。	false	

cephadm_bootstrap	説明	必須	デフォルト
ssh_user	ホストへの **cephadm** ssh に使用する SSH ユーザー。	false	
ssh_config	**cephadm** SSH クライアントの SSH 設定ファイルのパス。	false	
allow_fqdn_hostname	完全修飾ドメイン名 (FQDN) であるホスト名を許可します。	false	false
cluster_network	クラスターのレプリケーション、リカバリー、およびハートビートに使用するサブネット。	false	

表16.2 **ceph_orch_host モジュールで使用可能なオプション**

ceph_orch_host	説明	必須	デフォルト
fsid	対話する Ceph クラスターの FSID。	false	
image	使用する Ceph コンテナーイメージ。	false	
name	追加、削除、または更新するホストの名前。	true	
address	ホストの IP アドレス。	**state** が **present** である場合は true。	
set_admin_label	指定したホストに **_admin** ラベルを設定します。	false	false
labels	ホストに適用するラベルのリスト。	false	[]

ceph_orch_host	説明	必須	デフォルト
state	**present** に設定すると、name で指定された **name** が存在することが保証されます。**absent** に設定すると、**name** で指定されたホストが削除されます。**drain** に設定すると、**name** で指定されたホストからすべてのデーモンを削除するようにスケジュールされます。	false	あり

表16.3 ceph_config モジュールで利用可能なオプション

ceph_config	説明	必須	デフォルト
fsid	対話する Ceph クラスターの FSID。	false	
image	使用する Ceph コンテナーイメージ。	false	
action	**option** で指定されたパラメーターを **set** または **get** するかどうか。	false	set
who	設定を設定するデーモン。	true	
option	**set** または **get** するパラメーターの名前。	true	
値	設定するパラメーターの値。	アクションが **set** である場合は true	

表16.4 ceph_orch_apply モジュールで使用可能なオプション

ceph_orch_apply	説明	必須
fsid	対話する Ceph クラスターの FSID。	false
image	使用する Ceph コンテナーイメージ。	false

ceph_orch_apply	説明	必須
spec	適用するサービス仕様。	true

表16.5 **ceph_orch_daemon モジュールで使用可能なオプション**

ceph_orch_daemon	説明	必須
fsid	対話する Ceph クラスターの FSID。	false
image	使用する Ceph コンテナーイメージ。	false
state	**name** で指定されたサービスの望ましい状態。	true **started** の場合、サービスが確実に開始されます。 **stopped** の場合、サービスが確実に停止されます。 **restarted** の場合、サービスが再起動されます。
daemon_id	サービスの ID。	true
daemon_type	サービスのタイプ。	true

表16.6 **cephadm_registry_login モジュールで利用可能なオプション**

cephadm_registry_login	説明	必須	デフォルト
state	レジストリーのログインまたはログアウト。	false	login
docker	**podman** の代わりに **docker** を使用します。	false	
registry_url	カスタムレジストリーの URL。	false	
registry_username	カスタムレジストリーのユーザー名。	**state** が **login** の場合は **true**。	
registry_password	カスタムレジストリーのパスワード。	**state** が **login** の場合は **true**。	

cephadm_registry_login	説明	必須	デフォルト
registry_json	JSON ファイルへのパス。このファイルは、このタスクを実行する前にリモートホストに存在している必要があります。このオプションは現在サポートされていません。		

16.3. CEPHADM_BOOTSTRAP および CEPHADM_REGISTRY_LOGIN モジュールを使用したストレージクラスターのブートストラップ

ストレージ管理者は、Ansible Playbook で **cephadm_bootstrap** および **cephadm_registry_login** モジュールを使用して、Ansible を使用してストレージクラスターをブートストラップできます。

前提条件

- 最初の Ceph Monitor コンテナーの IP アドレス。これはストレージクラスターの最初のノードの IP アドレスでもあります。
- **registry.redhat.io** へのログインアクセス。
- 少なくとも 10 GB の空き容量がある **/var/lib/containers/**。
- **Ansible-core** が AppStream にバンドルされている Red Hat Enterprise Linux 9.0 以降。
- Ansible 管理ノードへの **cephadm-ansible** パッケージのインストール。
- パスワードなしの SSH がストレージクラスター内のすべてのホストに設定されます。
- ホストは CDN に登録されます。

手順

1. Ansible 管理ノードにログインします。
2. Ansible 管理ノードの **/usr/share/cephadm-ansible** ディレクトリーに移動します。

   例

   ```
   [ceph-admin@admin ~]$ cd /usr/share/cephadm-ansible
   ```

3. **hosts** ファイルを作成し、ホスト、ラベルを追加し、ストレージクラスター内の最初のホストの IP アドレスを監視します。

   構文

   ```
   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']"
```

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. プリフライト Playbook を実行します。

構文

```
ansible-playbook -i INVENTORY_FILE cephadm-preflight.yml --extra-vars
"ceph_origin=rhcs"
```

例

```
[ceph-admin@admin cephadm-ansible]$ ansible-playbook -i hosts cephadm-preflight.yml --
extra-vars "ceph_origin=rhcs"
```

5. クラスターをブートストラップする Playbook を作成します。

構文

```
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
```

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. Playbook を実行します。

構文

```
ansible-playbook -i INVENTORY_FILE PLAYBOOK_FILENAME.yml -vvv
```

例

```
[ceph-admin@admin cephadm-ansible]$ ansible-playbook -i hosts bootstrap.yml -vvv
```

検証

- Playbook を実行した後、Ansible の出力を確認します。

16.4. CEPH_ORCH_HOST モジュールを使用したホストの追加または削除

ストレージ管理者は、Ansible Playbook の **ceph_orch_host** モジュールを使用して、ストレージクラスター内のホストを追加および削除できます。

前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。

- ノードを CDN に登録して、サブスクリプションを割り当てます。
- ストレージクラスター内のすべてのノードへの sudo アクセスおよびパスワードなしの SSH アクセスのある Ansible ユーザー。
- Ansible 管理ノードへの **cephadm-ansible** パッケージのインストール。
- 新しいホストには、ストレージクラスターの公開 SSH キーがあります。ストレージクラスターの公開 SSH キーを新しいホストにコピーする方法の詳細については、**Red Hat Ceph Storage インストールガイド** の **ホストの追加** を参照してください。

手順

1. 次の手順を使用して、新しいホストをクラスターに追加します。
   a. Ansible 管理ノードにログインします。
   b. Ansible 管理ノードの **/usr/share/cephadm-ansible** ディレクトリーに移動します。

      例

      ```
      [ceph-admin@admin ~]$ cd /usr/share/cephadm-ansible
      ```

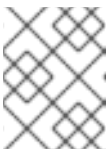
   c. 新しいホストとラベルを Ansible インベントリーファイルに追加します。

      構文

      ```
      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']"
      ```

      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']"
      ```

      注記

      新しいホストを Ansible インベントリーファイルに追加し、ホストでプリフライト Playbook を実行している場合は、ステップ 3 に進みます。

d. **--limit** オプションを指定して、プリフライト Playbook を実行します。

構文

```
ansible-playbook -i INVENTORY_FILE cephadm-preflight.yml --extra-vars
"ceph_origin=rhcs" --limit NEWHOST
```

例

```
[ceph-admin@admin cephadm-ansible]$ ansible-playbook -i hosts cephadm-preflight.yml
--extra-vars "ceph_origin=rhcs" --limit host02
```

プリフライト Playbook は、新しいホストに **podman**、**lvm2**、**chronyd**、および **cephadm** をインストールします。インストールが完了すると、**cephadm** は **/usr/sbin/** ディレクトリーに配置されます。

e. 新しいホストをクラスターに追加する Playbook を作成します。

構文

```
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 }}"
```

注記

デフォルトでは、Ansible は Playbook の **hosts** 行に一致するホストですべてのタスクを実行します。**ceph orch** コマンドは、管理キーリングと Ceph 設定ファイルを含むホストで実行する必要があります。**delegate_to** キーワードを使用して、クラスター内の管理ホストを指定します。

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 }}"
```

この例では、Playbook は新しいホストをクラスターに追加し、ホストの現在のリストを表示します。

f. Playbook を実行して、追加のホストをクラスターに追加します。

構文

```
ansible-playbook -i INVENTORY_FILE PLAYBOOK_FILENAME.yml
```

例

```
[ceph-admin@admin cephadm-ansible]$ ansible-playbook -i hosts add-hosts.yml
```

2. 次の手順を使用して、クラスターからホストを削除します。

a. Ansible 管理ノードにログインします。

b. Ansible 管理ノードの **/usr/share/cephadm-ansible** ディレクトリーに移動します。

例

```
[ceph-admin@admin ~]$ cd /usr/share/cephadm-ansible
```

c. クラスターからホストを削除する Playbook を作成します。

構文

```
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 }}"
```

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 }}"
```

この例では、playbook タスクは **host07** 上のすべてのデーモンをドレインし、クラスターからホストを削除し、ホストの現在のリストを表示します。

d. Playbook を実行して、クラスターからホストを削除します。

構文

```
ansible-playbook -i INVENTORY_FILE PLAYBOOK_FILENAME.yml
```

例

```
[ceph-admin@admin cephadm-ansible]$ ansible-playbook -i hosts remove-hosts.yml
```

検証

- クラスター内のホストの現在のリストを表示する Ansible タスクの出力を確認します。

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. CEPH_CONFIG モジュールを使用した設定オプションの設定

ストレージ管理者は、**ceph_config** モジュールを使用して Red Hat Ceph Storage 設定オプションを設定または取得できます。

前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。
- ストレージクラスター内のすべてのノードへの sudo アクセスおよびパスワードなしの SSH アクセスのある Ansible ユーザー。

- Ansible 管理ノードへの **cephadm-ansible** パッケージのインストール。
- Ansible インベントリーファイルには、クラスターと管理ホストが含まれている。

手順

1. Ansible 管理ノードにログインします。
2. Ansible 管理ノードの **/usr/share/cephadm-ansible** ディレクトリーに移動します。

   例

   ```
   [ceph-admin@admin ~]$ cd /usr/share/cephadm-ansible
   ```

3. 設定を変更して Playbook を作成します。

   構文

   ```
   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 }}"
   ```

   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 }}"
```

この例では、Playbook は最初に **mon_allow_pool_delete** オプションを **false** に設定します。その後、Playbook は現在の **mon_allow_pool_delete** 設定を取得し、その値を Ansible 出力に表示します。

4. Playbook を実行します。

構文

```
ansible-playbook -i INVENTORY_FILE _PLAYBOOK_FILENAME.yml
```

例

```
[ceph-admin@admin cephadm-ansible]$ ansible-playbook -i hosts change_configuration.yml
```

検証

- Playbook タスクからの出力を確認します。

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
```

関連情報

- 設定オプションの詳細は、Red Hat Ceph Storage 設定ガイド を参照してください。

16.6. CEPH_ORCH_APPLY モジュールを使用したサービス仕様の適用

ストレージ管理者は、Ansible Playbook の **ceph_orch_apply** モジュールを使用して、ストレージクラスターにサービス仕様を適用できます。サービス仕様は、Ceph サービスのデプロイに使用されるサービス属性および設定を指定するデータ構造です。サービス仕様を使用し

て、**mon**、**crash**、**mds**、**mgr**、**osd**、**rdb**、または **rbd-mirror** などの Ceph サービスタイプをデプロイできます。

前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。
- ストレージクラスター内のすべてのノードへの sudo アクセスおよびパスワードなしの SSH アクセスのある Ansible ユーザー。
- Ansible 管理ノードへの **cephadm-ansible** パッケージのインストール。
- Ansible インベントリーファイルには、クラスターと管理ホストが含まれている。

手順

1. Ansible 管理ノードにログインします。
2. Ansible 管理ノードの **/usr/share/cephadm-ansible** ディレクトリーに移動します。

   例

   ```
   [ceph-admin@admin ~]$ cd /usr/share/cephadm-ansible
   ```

3. サービス仕様を使用して Playbook を作成します。

   構文

   ```
   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:
   ```

   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
```

この例では、Playbook はラベル **osd** を持つすべてのホストに Ceph OSD サービスをデプロイします。

4. Playbook を実行します。

構文

```
ansible-playbook -i INVENTORY_FILE _PLAYBOOK_FILENAME.yml
```

例

```
[ceph-admin@admin cephadm-ansible]$ ansible-playbook -i hosts deploy_osd_service.yml
```

検証

- Playbook タスクからの出力を確認します。

関連情報

- サービス仕様オプションの詳細は、Red Hat Ceph Storage Operations Guide を参照してください。

16.7. CEPH_ORCH_DAEMON モジュールを使用した CEPH デーモンの状態の管理

ストレージ管理者は、Ansible Playbook の **ceph_orch_daemon** モジュールを使用して、ホストで Ceph デーモンを開始、停止、および再起動できます。

前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。
- ストレージクラスター内のすべてのノードへの sudo アクセスおよびパスワードなしの SSH アクセスのある Ansible ユーザー。
- Ansible 管理ノードへの **cephadm-ansible** パッケージのインストール。
- Ansible インベントリーファイルには、クラスターと管理ホストが含まれている。

手順

1. Ansible 管理ノードにログインします。
2. Ansible 管理ノードの **/usr/share/cephadm-ansible** ディレクトリーに移動します。

   例

   ```
   [ceph-admin@admin ~]$ cd /usr/share/cephadm-ansible
   ```

3. デーモンの状態が変化する Playbook を作成します。

   構文

   ```
   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
   ```

   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
   ```

   この例では、Playbook は ID **0** で OSD を開始し、ID が **host02** の Ceph Monitor を停止します。

4. Playbook を実行します。

構文

```
ansible-playbook -i INVENTORY_FILE _PLAYBOOK_FILENAME.yml
```

例

```
[ceph-admin@admin cephadm-ansible]$ ansible-playbook -i hosts restart_services.yml
```

検証

- Playbook タスクからの出力を確認します。

付録A MCLOCK 設定オプション

このセクションには、mClock 設定オプションのリストが含まれています。

osd_mclock_profile

説明

バックグラウンドリカバリー、**backfill**、**pg scrub**、**snap trim** 、**client op**、**pg deletion** など、さまざまなクラスに属する操作に基づいてサービス品質 (QoS) を提供するために使用する mClock プロファイルのタイプを設定します。
ビルトイン プロファイルが有効になると、下位レベルの mClock リソース制御パラメーター (予約、重み、制限) と一部の Ceph 設定パラメーターが透過的に設定されます。これは、**カスタム** プロファイルには適用されません。

型

String

デフォルト

balanced

有効な選択肢

balanced, **high_recovery_ops**, **high_client_ops**, **custom**

osd_mclock_max_capacity_iops_hdd

説明

回転メディアの OSD ごとに考慮される最大ランダム書き込み IOPS 容量を 4 KiB ブロックサイズに設定します。dm Clock プロファイルを有効にする場合の QoS 計算に貢献します。**osd_op_queue = mlock_scheduler** の場合のみ考慮されます。

型

浮動小数点 (Float)

デフォルト

315.0

osd_mclock_max_capacity_iops_ssd

説明

ソリッドステートメディアの OSD ごとに考慮される最大ランダム書き込み IOPS 容量を 4 KiB ブロックサイズに設定します。

型

浮動小数点 (Float)

デフォルト

21500.0

osd_mclock_cost_per_byte_usec_ssd

説明

SDD の OSD ごとに考慮すべきバイトあたりのコストをマイクロ秒単位で示します。dm Clock プロファイルを有効にする場合の QoS 計算に貢献します。**osd_op_queue = mlock_scheduler** の場合のみ考慮されます。

型

浮動小数点 (Float)

デフォルト

0.011

osd_mclock_max_sequential_bandwidth_hdd

説明

基礎となるデバイスタイプが回転メディアである OSD について考慮する最大シーケンシャル帯域幅をバイト単位で示します。これは、QoS 計算で使用されるコスト係数を導出するために m Clock スケジューラーによって考慮されます。**osd_op_queue = mlock_scheduler** の場合のみ考慮されます。

型

サイズ

デフォルト

150_M

osd_mclock_max_sequential_bandwidth_ssd

説明

基礎となるデバイスタイプがソリッドステートメディアである OSD について考慮する最大シーケンシャル帯域幅をバイト単位で示します。これは、QoS 計算で使用されるコスト係数を導出するために m Clock スケジューラーによって考慮されます。**osd_op_queue = mlock_scheduler** の場合のみ考慮されます。

型

サイズ

デフォルト

1200_M

osd_mclock_force_run_benchmark_on_init

説明

OSD の初期化時または起動時に OSD ベンチマークが強制的に実行されます。

型

Boolean

デフォルト

False

関連項目

osd_mclock_max_capacity_iops_hdd, **osd_mclock_max_capacity_iops_ssd**

osd_mclock_skip_benchmark

説明

このオプションを設定すると、OSD の初期化時または起動時に OSD ベンチマークの実行が省略されます。

型

Boolean

デフォルト

False

関連項目

osd_mclock_max_capacity_iops_hdd, **osd_mclock_max_capacity_iops_ssd**

osd_mclock_override_recovery_settings

説明

このオプションを設定すると、**osd_recovery_max_active_hdd**、**osd_recovery_max_active_ssd**、および **osd_max_backfills** オプションで定義されているように、mClock スケジューラーのリカバリーまたはバックフィルの制限をオーバーライドできます。

型

Boolean

デフォルト

False

関連項目

osd_recovery_max_active_hdd, **osd_recovery_max_active_ssd**, **osd_max_backfills**

osd_mclock_iops_capacity_threshold_hdd

説明

これは、4KiB ブロックサイズでの IOPS 容量のしきい値を示します。これを超えると、HDD の OSD の Ceph OSD ベンチ結果が無視されます。

型

浮動小数点 (Float)

デフォルト

500.0

osd_mclock_iops_capacity_threshold_ssd

説明

これは、4KiB ブロックサイズでの IOPS 容量のしきい値を示します。これを超えると、SSD の OSD の Ceph OSD ベンチ結果が無視されます。

型

浮動小数点 (Float)

デフォルト

80000.0

osd_mclock_scheduler_client_res

説明

各クライアントに予約されているデフォルトの I/O 比率です。デフォルト値の **0** は、可能な限り最も低い予約を指定します。0 より大きく 1.0 までの値は、OSD の最大 IOPS 容量の一部として各クライアントに予約する最小 IO 比率を指定します。

型

float

デフォルト

0

min

0

max

1.0

osd_mclock_scheduler_client_wgt

説明

予約を超えた場合の、各クライアントのデフォルトの I/O 割り当てです。

型

符号なしの整数

デフォルト

1

osd_mclock_scheduler_client_lim

説明

予約を超えた場合の、各クライアントのデフォルトの I/O 制限です。デフォルト値の **0** は、制限の適用を指定しません。これは、各クライアントが OSD の可能な最大 IOPS 容量を使用できることを意味します。0 より大きく 1.0 までの値は、OSD の最大 IOPS 容量の一部として各クライアントが受け取る予約に対する IO の上限を指定します。

型

float

デフォルト

0

min

0

max

1.0

osd_mclock_scheduler_background_recovery_res

説明

バックグラウンドリカバリー用に予約されているデフォルトの I/O 比率です。デフォルト値の 0 は、可能な限り最も低い予約を指定します。0 より大きく 1.0 までの値は、OSD の最大 IOPS 容量の一部として、バックグラウンド回復操作用に予約する最小 IO 比率を指定します。

型

float

デフォルト

0

min

0

max

1.0

osd_mclock_scheduler_background_recovery_wgt

説明

予約を超えた場合の、各バックグラウンドリカバリーの I/O 割り当てを示します。

型

符号なしの整数

デフォルト

1

osd_mclock_scheduler_background_recovery_lim

説明

予約を超えた場合の、バックグラウンドリカバリーの I/O 制限を示します。デフォルト値の 0 は、制限の適用を指定しません。これは、バックグラウンド回復操作で OSD の可能な最大 IOPS 容量を使用できることを意味します。0 より大きく 1.0 までの値は、バックグラウンド回復操作が OSD の最大 IOPS 容量の一部として受け取る予約に対する IO の上限を指定します。

型

float

デフォルト

0

min

0

max

1.0

osd_mclock_scheduler_background_best_effort_res

説明

バックグラウンド **best_effort** 用に予約されているデフォルトの I/O 比率を示します。デフォルト値の 0 は、可能な限り最も低い予約を指定します。0 より大きく 1.0 までの値は、OSD の最大 IOPS 容量の一部として、バックグラウンドの best_effort 操作用に予約する最小 IO 比率を指定します。

型

float

デフォルト

0

min

0

max

1.0

osd_mclock_scheduler_background_best_effort_wgt

説明

予約を超えた場合の、各バックグラウンド **best_effort** の I/O 割り当てを示します。

型

符号なしの整数

デフォルト

1

osd_mclock_scheduler_background_best_effort_lim

説明

予約を超えた場合の、バックグラウンド **best_effort** の I/O 制限を示します。デフォルト値の 0 は、制限の適用なしを指定します。これは、バックグラウンドの best_effort 操作が OSD の可能な最大 IOPS 容量を使用できることを意味します。0 より大きく 1.0 までの値は、バックグラウンドの best_effort 操作が OSD の最大 IOPS 容量の一部として受け取る予約に対する IO の上限を指定します。

型

float

デフォルト

0

min

0

max

1.0

関連情報

osd_op_queue オプションの詳細は、Object Storage Daemon (OSD) の設定オプションを参照してください。