Red Hat Ceph Storage 5

管理ガイド

Red Hat Ceph Storage の管理

Last Updated: 2022-05-21

Red Hat Ceph Storage 5 管理ガイド

Red Hat Ceph Storage の管理

Enter your first name here. Enter your surname here.
Enter your organisation's name here. Enter your organisational division here.
Enter your email address here.

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 種類のデーモンで構成されます。

- 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。

関連情報

- 『Red Hat Ceph Storage インストールガイド』

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. 前提条件

- Red Hat Ceph Storage ソフトウェアのインストール

2.2. 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 の使用の詳細については、Red Hat Enterprise Linux 8 **基本的なシステム設定の構成** ガイドの **systemd の概要** の章、および **systemctl を使用したシステムサービスの管理** の章を参照してください。

2.3. すべての CEPH デーモンの開始、停止、および再起動

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
   ```

2.4. 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-5-
   rhel8:latest           c88a5d60f510
   grafana                     1/1  4m ago     4M   count:1    registry.redhat.io/rhceph-alpha/rhceph-5-
   dashboard-rhel8:latest  bd3d7748747b
   mgr                         2/2  4m ago     4M   count:2    registry.redhat.io/rhceph-alpha/rhceph-5-
   rhel8:latest           c88a5d60f510
   mon                         2/2  4m ago     10w  count:2    registry.redhat.io/rhceph-alpha/rhceph-5-
   rhel8: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-5-rhel8: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-5-rhel8: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お勧めします。

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
```

2.5. 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、以下の形式で構成される **root** で **journalctl** コマンドを実行します。

構文

```
journalctl -u ceph SERVICE_ID
```

```
[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。

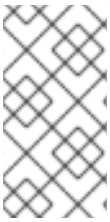
2. 最近のジャーナルエントリーのみを表示するには、**-f** オプションを使用します。

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

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

Ceph クラスターの電源をオフにしてリブートするには、以下の手順を実施します。

前提条件

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

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他のクライアントで RBD イメージおよび RADOS Gateway を使用しないようにします。
2. 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。
3. Ceph File System (**CephFS**) を使用する場合は、**CephFS** クラスターを停止する必要があります。

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
```

4. **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。クライアントキーリングが設定されたノードで以下のコマンドを実行します。Ceph Monitor または OpenStack コントローラーノードの例を以下に示します。

   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
```

5. 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
```

6. 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
```

Red Hat Ceph Storage クラスターのリブート

1. 管理ノードの電源を入れます。
2. 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
```

3. 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
```

4. 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。

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** フラグの設定を解除します。クライアントキーリングが設定されたノードで以下のコマンドを実行します。Ceph Monitor または OpenStack コントローラーノードの例を以下に示します。

   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
```

6. Ceph File System (**CephFS**) を使用する場合、**CephFS** クラスターは、**joinable** フラグを **true** に設定して再び起動する必要があります。

   構文

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

   例

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

7. 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**)。クライアントキーリングが設定されたノードで **ceph status** を実行します。たとえば、クラスターが正常であることを確認する Ceph Monitor または OpenStack コントローラーノードなど。

関連情報

- Ceph のインストールに関する詳細は、『Red Hat Ceph Storage Installation Guide』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. 前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。

3.2. 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 Dashboard 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ース、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。

3.2.1. 前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。

3.2.2. Ceph コマンドラインインターフェースの対話形式の使用

ceph コマンドラインユーティリティーを使用して、Ceph ストレージクラスターと対話的にインターフェースで接続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. インタラクティブモードで **ceph** ユーティリティーを実行するには、以下を行います。

   構文

   ```
   podman exec -it ceph-mon-MONITOR_NAME /bin/bash
   ```

   置き換え

- **MONITOR_NAME** は、Ceph Monitor コンテナーの名前に置き換えます。この名前は、**podman ps** コマンドを実行して見つけます。

例

```
[root@host01 ~]# podman exec -it ceph-499829b4-832f-11eb-8d6d-001a4a000635-mon.host01 /bin/bash
```

この例では、**mon.host01** で対話的なターミナルセッションを開き、ここで Ceph の対話型シェルを起動することができます。

3.2.3. 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。

   - クラスター 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
  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。

3.2.4. 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ースを使用して、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
```

3.2.5. 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。

3.2.6. 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。**ceph df** コマンドまたは **ceph df detail** コマンドのいずれかを実行できます。

例

```
[ceph: root@host01 /]# ceph df
RAW STORAGE:
    CLASS     SIZE       AVAIL      USED        RAW USED     %RAW USED
    hdd       90 GiB     84 GiB     100 MiB      6.1 GiB         6.78
    TOTAL     90 GiB     84 GiB     100 MiB      6.1 GiB         6.78

POOLS:
    POOL                    ID     STORED      OBJECTS     USED       %USED     MAX AVAIL
    .rgw.root               1     1.3 KiB           4     768 KiB         0        26 GiB
    default.rgw.control             2         0 B           8         0 B         0        26 GiB
    default.rgw.meta                3     2.5 KiB          12     2.1 MiB         0        26 GiB
    default.rgw.log                 4     3.5 KiB         208     6.2 MiB         0        26 GiB
```

```
    default.rgw.buckets.index      5     2.4 KiB          33     2.4 KiB        0        26 GiB
    default.rgw.buckets.data       6     9.6 KiB          15     1.7 MiB        0        26 GiB
    testpool                      10       231 B           5     384 KiB        0        40 GiB
```

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。

例

```
[ceph: root@host01 /]# ceph df detail
RAW STORAGE:
    CLASS     SIZE       AVAIL      USED        RAW USED     %RAW USED
    hdd       90 GiB     84 GiB     100 MiB      6.1 GiB          6.78
    TOTAL     90 GiB     84 GiB     100 MiB      6.1 GiB          6.78

POOLS:
    POOL                        ID     STORED      OBJECTS     USED        %USED     MAX AVAIL
QUOTA OBJECTS     QUOTA BYTES     DIRTY     USED COMPR     UNDER COMPR
    .rgw.root                    1     1.3 KiB          4     768 KiB         0       26 GiB     N/A               N/A
4            0 B             0 B
    default.rgw.control          2         0 B          8         0 B         0       26 GiB     N/A               N/A
8            0 B             0 B
    default.rgw.meta             3     2.5 KiB         12     2.1 MiB         0       26 GiB     N/A               N/A
12           0 B             0 B
    default.rgw.log              4     3.5 KiB        208     6.2 MiB         0       26 GiB     N/A               N/A
208          0 B             0 B
    default.rgw.buckets.index    5     2.4 KiB         33     2.4 KiB         0       26 GiB     N/A               N/A
33           0 B             0 B
    default.rgw.buckets.data     6     9.6 KiB         15     1.7 MiB         0       26 GiB     N/A               N/A
15           0 B             0 B
    testpool                    10       231 B          5     384 KiB         0       40 GiB     N/A               N/A
5            0 B             0 B
```

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に使用するストレージ容量の概要を説明します。

- **CLASS:** 使用されるデバイスのタイプ。
- **SIZE:** ストレージクラスターが管理する全ストレージ容量。
  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。
- **USD:** 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に消費される、ストレージクラスター内の使用済み領域の量。
  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:** USED 領域の合計と、BlueStore パーティション **db** および **wal** に割り当てられた領域の合計。
- **% 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。

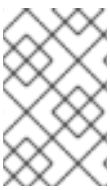
出力の POOLS セクションは、プールの一覧と、各プールの概念的な使用目的を提供します。このセクションの出力には、レプリカ、クローン、またはスナップショットを **反映しません**。たとえば、1 MB のデータでオブジェクトを保存する場合、概念的な使用量は 1 MB になりますが、実際の使用量は、**size = 3** のクローンやスナップショットなどのレプリカ数によっては 3 MB 以上になる場合があります。

- **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。
- **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

- 詳細については、「Ceph のデータ使用量の計算方法」を参照してください。
- 詳細については、「OSD の使用状況の統計について」を参照してください。

3.2.7. 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

- 詳細については、「**Ceph のデータ使用量の計算方法**」を参照してください。
- 詳細については、「**OSD の使用状況の統計について**」を参照してください。

- 詳細については、『**Red Hat Ceph Storage Storage Strategies Guide**』の「CRUSH Weights」を参照してください。

3.2.8. 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ースから 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

   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
```

3.2.9. 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

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": "{}"
            }
        ]
    }
}
```


3.2.10. Ceph 管理ソケットの使用

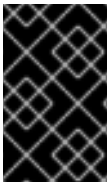
管理ソケットを使用して、UNIX ソケットファイルを使用して、指定したデーモンと直接対話します。たとえば、ソケットを使用すると以下を行うことができます。

- ランタイム時に Ceph 設定を一覧表示します。
- 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一覧表示するには、**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",
         "dump_historic_ops": "dump_historic_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 perfcounters value",
    "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 perfcounters schema",
    "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. **osd.2** という名前の Ceph OSD のステータスを表示するには、以下のコマンドを実行します。

   例

   ```
   [ceph: root@host01 /]# ceph daemon /var/run/ceph/ceph-osd.2.asok status
   ```

5. Ceph プロセスのソケットファイルの一覧を表示するには、以下のコマンドを実行します。

   例

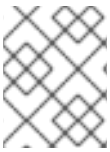
   ```
   [ceph: root@host01 /]# ls /var/run/ceph
   ```

関連情報

- 詳細は、『Red Hat Ceph Storage Troubleshooting Guide』を参照してください。

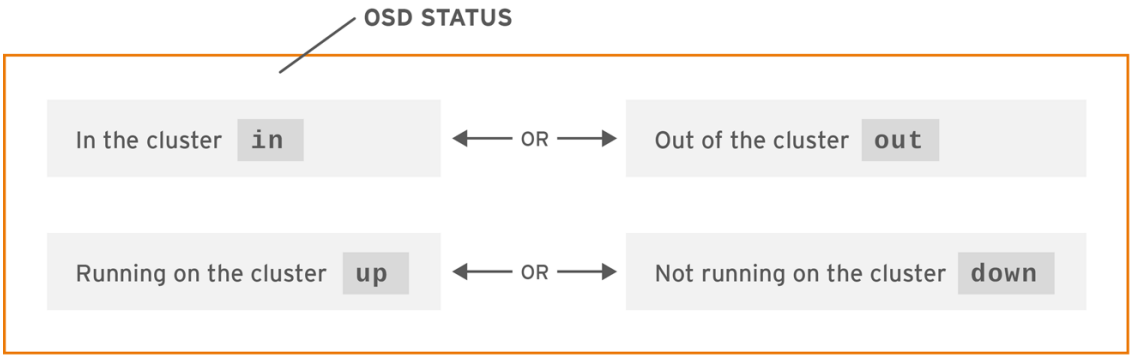
3.2.11. Ceph OSD のステータスについて

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、または **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

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

- 詳細については、『Red Hat Ceph Storage Dashboard Guide』を参照してください。

3.3. 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。

3.3.1. 前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。

3.3.2. 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、動作セットの構成を変更し、「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 がその役割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、または問題がある場合に、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. 配置グループの一覧を取得するには、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** 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
   ```

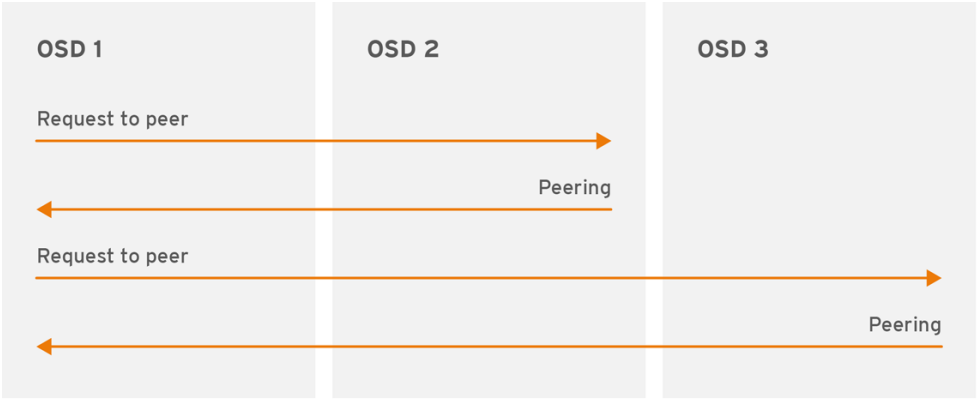
注記

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。

3.3.3. 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



3.3.4. 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。

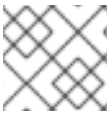
すべての配置グループのステータスを表示するには、以下を実行します。

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構成され、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
```

- 配置グループの一覧を取得するには、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
```

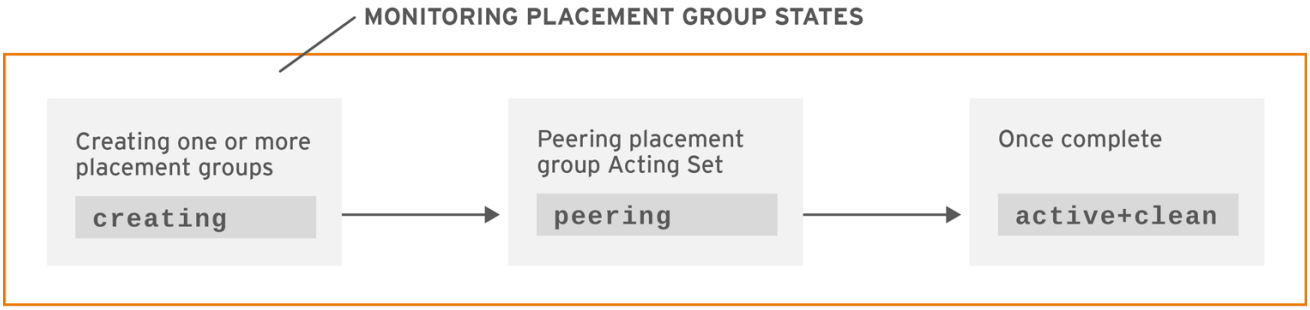
```
        },
        "stats": {
            "version": "0'0",
            "reported_seq": "2989",
            "reported_epoch": "2449",
            "state": "active+clean",
            "last_fresh": "2021-06-18T05:16:59.401080+0000",
            "last_change": "2021-06-17T01:32:03.764162+0000",
            "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 Configuration Guide** の OSD Object storage daemon configuratopn options の **Object Storage Daemon (OSD) configuration options** を参照してください。

3.3.5. 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

3.3.6. 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。

3.3.7. 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。

3.3.8. 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。

3.3.9. 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役割を担います。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。

3.3.10. 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。

3.3.11. 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。

3.3.12. 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。

3.3.13. 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。

3.3.14. 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。

3.3.15. 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。

3.3.16. 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
   ```

3.3.17. 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 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。

4.1. 前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。

4.2. 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

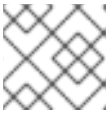
   ```
   [ceph: root@host01 /]# ceph osd unset noout
   ```

フラグ	詳細
noin	OSD が、クラスター **内** での扱いになるのを防ぎます。

フラグ	詳細
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。
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。

4.3. 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。

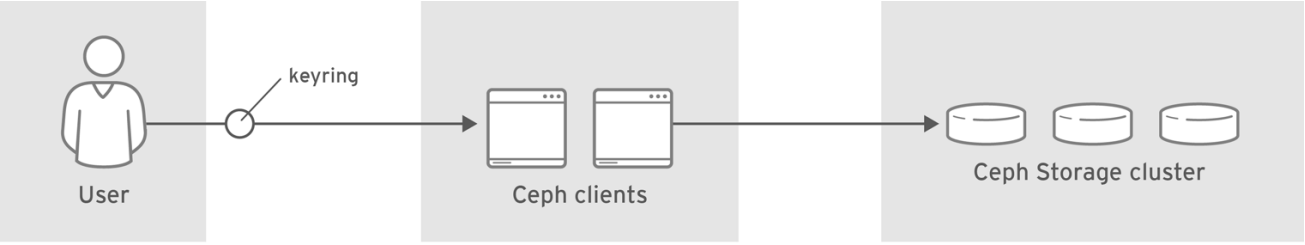
- **norecover**: OSD を置き換える必要がえあり、ディスクをホットスワップする間に PG を別の OSD に復元しないようする場合は、他の OSD のセットが他の OSD に新しいセットをコピーしないように、**norecover** も設定できます。

- **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。

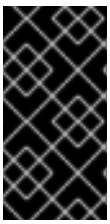
第5章 CEPH ユーザー管理

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。



5.1. 前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。
- Ceph Monitor または Ceph クライアントノードへのアクセス



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。

5.2. 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 で構成されるピリオド (.) 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。

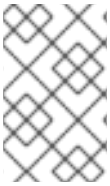
プール

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 Configuration Guide』を参照してください。

5.3. 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。

5.3.1. 前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。
- Ceph Monitor または Ceph クライアントノードへのアクセス

5.3.2. Ceph ユーザーの一覧表示

コマンドラインインターフェースを使用して、ストレージクラスター内のユーザーを一覧表示できます。

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一覧表示するには、以下を実行します。

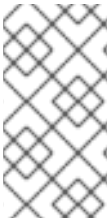
   例

```
[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: AQD0RE9gkxA4ExAAFXp3pLJWdIhsyTe2ZR6Ilw==
 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。

5.3.3. Ceph ユーザー情報の表示

コマンドラインインターフェースを使用して、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。

5.3.4. 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。

5.3.5. 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。

5.3.6. Ceph ユーザーの削除

コマンドラインインターフェースを使用して、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
   ```

5.3.7. Ceph ユーザーキーの出力

コマンドラインインターフェースを使用して、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

1. ユーザーの認証キーを標準出力に出力するには、以下を実行します。

   構文

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

   例

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

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

2. ユーザーのキーを出力すると、クライアントソフトウェアにユーザーのキー (例] **libvirt**) を設定する必要がある場合に便利です。

   構文

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

例

```
[ceph: root@host01 /]#  mount -t ceph host02:/ceph -o name=client.user,secret=`ceph auth
print-key client.user`
```

第6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。

6.1. 前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。

6.2. パフォーマンスベースライン

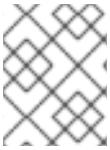
ジャーナル、ディスク、ネットワークのスループットを含む OSD には、比較すべきパフォーマンスベースラインがあるはずです。ベースラインのパフォーマンスデータと Ceph のネイティブツールのデータを比較することで、潜在的なチューニング効果を特定することができます。Red Hat Enterprise Linux には、これらのタスクを実現するために利用可能なオープンソースコミュニティーツールが複数含まれています。

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。

6.3. 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
```

6.4. 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

1. **rbd** カーネルモジュールを読み込んでいない場合は読み込みます。

   例

```
[root@host01 ~]# modprobe rbd
```

2. **testbench** プールに 1 GB の **rbd** イメージファイルを作成します。

   例

   ```
   [root@host01 ~]# rbd create image01 --size 1024 --pool testbench
   ```

3. イメージファイルをデバイスファイルにマッピングします。

   例

   ```
   [root@host01 ~]# rbd map image01 --pool testbench --name client.admin
   ```

4. ブロックデバイスに **ext4** ファイルシステムを作成します。

   例

   ```
   [root@host01 ~]# mkfs.ext4 /dev/rbd/testbench/image01
   ```

5. 新しいディレクトリーを作成します。

   例

   ```
   [root@host01 ~]# mkdir /mnt/ceph-block-device
   ```

6. ブロックデバイスを **/mnt/ceph-block-device/** にマウントします。

   例

   ```
   [root@host01 ~]# mount /dev/rbd/testbench/image01 /mnt/ceph-block-device
   ```

7. ブロックデバイスに対して書き込みパフォーマンスの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** コマンドの詳細については、Red Hat Ceph Storage Block Device Guide の Ceph block devices の章を参照してください。

第7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ックを収集できます。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ックのコレクションです。このメトリックデータの収集、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。

7.1. 前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。

7.2. 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ー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一覧です。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

7.3. 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、利用可能なメトリックを出力します。各メトリックには、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. メトリック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7.1 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)

ビット	意味
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。

7.4. 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ック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 メトリックの簡単な説明を表示するには、Ceph Monitor メトリックの表を参照してください。

7.5. 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 メトリックの簡単な説明を表示するには、Ceph OSD メトリックの表を参照してください。

7.6. CEPH MONITOR メトリック

- クラスターメトリックテーブル
- 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ックテーブル
- Paxos Metrics テーブル
- スロットルメトリックテーブル

表7.2 クラスターメトリックテーブル

コレクション名	メトリック名	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

コレクション名	メトリック名	ビットフィールド値	簡単な説明
	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
	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7.3 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	メトリック名	ビットフィールド値	簡単な説明
leveldb	**leveldb_get**	10	取得

コレクション名	メトリック名	ビットフィールド値	簡単な説明
	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7.4 一般的なモニターメトリックテーブル

コレクション名	メトリック名	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7.5 Paxos Metrics テーブル

コレクション名	メトリック名	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。

コレクション名	メトリック名	ビットフィールド値	簡単な説明
	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
	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

コレクション名	メトリック名	ビットフィールド値	簡単な説明
	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
	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7.6 スロットルメトリックテーブル

コレクション名	メトリック名	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

コレクション名	メトリック名	ビットフィールド値	簡単な説明
	put_sum	10	データを送る
	wait	5	待機レイテンシー

7.7. CEPH OSD メトリック

- ライトバックスロットルメトリックテーブル
- 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ックテーブル
- リカバリー状態のメトリックテーブル
- OSD スロットルのメトリクステーブル

表7.7 ライトバックスロットルメトリックテーブル

コレクション名	メトリック名	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7.8 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	メトリック名	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

コレクション名	メトリック名	ビットフィールド値	簡単な説明
	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7.9 Objecter Metrics テーブル

コレクション名	メトリック名	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

コレクション名	メトリック名	ビットフィールド値	簡単な説明
	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
	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

コレクション名	メトリック名	ビットフィールド値	簡単な説明
	linger_send	10	送信された linger 操作
	linger_resend	10	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
	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7.10 読み出し操作および書き込み操作のメトリックテーブル

コレクション名	メトリック名	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
	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

コレクション名	メトリック名	ビットフィールド値	簡単な説明
	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)
	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

コレクション名	メトリック名	ビットフィールド値	簡単な説明
	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
	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

コレクション名	メトリック名	ビットフィールド値	簡単な説明
	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
	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

表7.11 リカバリー状態のメトリックテーブル

コレクション名	メトリック名	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
	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

コレクション名	メトリック名	ビットフィールド値	簡単な説明
	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
	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7.12 OSD スロットルのメトリクステーブル

コレクション名	メトリック名	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

7.8. CEPH OBJECT GATEWAY メトリックス

- 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7.13 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	メトリック名	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	中止要求
	get	10	取得
	get_b	10	取得サイズ
	get_initial_lat	5	レイテンシーの取得

コレクション名	メトリック名	ビットフィールド値	簡単な説明
	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7.14 Objecter Metrics テーブル

コレクション名	メトリック名	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

コレクション名	メトリック名	ビットフィールド値	簡単な説明
	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
	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

コレクション名	メトリック名	ビットフィールド値	簡単な説明
	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	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
	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

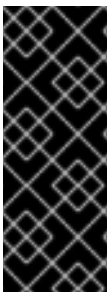
コレクション名	メトリック名	ビットフィールド値	簡単な説明
	map_full	10	受け取った完全な OSD マップ
	map_inc	10	受け取った増分 OSD マップ
	osd_sessions	2	セッションを開く
	osd_session_open	10	開いたセッション
	osd_session_close	10	閉じたセッション
	osd_laggy	2	Laggy OSD セッション

表7.15 Ceph Object Gateway スロットルメトリクステーブル

コレクション名	メトリック名	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

第8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。

8.1. 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。

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 (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。

8.2. CEPH BLUESTORE デバイス

本セクションでは、BlueStore バックエンドが使用するブロックデバイスを説明します。

BlueStore は、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。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ギガバイトストレージのみが存在する場合。Red Hat は、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。

8.3. CEPH BLUESTORE キャッシュ

BlueStore キャッシュ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

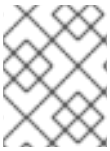
bluestore_default_buffered_write オプションを有効にする前に、Red Hat サポート にお問い合わせください。

8.4. 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。



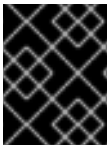
注記

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。

8.5. BLUESTORE_MIN_ALLOC_SIZE パラメータを使用した BLUESTORE の調整

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。

書き込み増幅のペナルティーを回避できるように **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お勧めしません。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 6144
   ```

   bluestore_min_alloc_size が 6144 バイトに設定されていることがわかります。これは 6 KiB に相当します。

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": "6144"
}
```

8.6. 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。

前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。
- BlueStore として設定されたオブジェクトストア。
- Red Hat Ceph Storage のコンテナー化されたデプロイメント。
- ノードにデプロイされた OSD ノード
- ノードへのルートレベルのアクセス。

手順

1. パッケージをダウンロードします。

   例

   ```
   [root@host01 ~]# yum config-manager --add-repo=http://download.eng.bos.redhat.com/rhel-8/composes/auto/ceph-5.0-rhel-8/latest-RHCEPH-5-RHEL-8/compose/OSD/x86_64/os/
   ```

2. **ceph-osd** パッケージをインストールします。このパッケージには、再シャード化に使用される **ceph-bluestore-tool** が含まれています。

   例

   ```
   [root@host01 ~]# dnf install -y ceph-osd
   ```

3. OSD サービスを停止するには、**systemctl** コマンドを実行して SERVICE_ID_OF_OSD を取得する必要があります。

   a. **systemctl** サービスを実行して、停止する必要のある OSD の **OSD の SERVICE_ID_OF_** を取得します。

      例

```
[root@host01 ~]# systemctl --type=service

ceph-4709608e-9089-11eb-bb5a-001a4a000740@osd.6.service
```

b. OSD サービスを停止します。

構文

```
systemctl stop SERVICE_ID_OF_OSD
```

例

```
[root@host01 ~]# systemctl stop ceph-4709608e-9089-11eb-bb5a-
001a4a000740@osd.6.service
```

4. OSD がデプロイされているディレクトリーに移動します。

構文

```
cd /var/lib/ceph/OSD_DIRECTORY/OSD_ID/
```

例

```
[root@host01 ~]# cd /var/lib/ceph/4709608e-9089-11eb-bb5a-001a4a000740/osd.6/
```

5. ディレクトリーにある **unit.run** ファイルをバックアップしてから変更してください。

例

```
[root@host01 osd.6]# cp /var/lib/ceph/4709608e-9089-11eb-bb5a-
001a4a000740/osd.6/unit.run /var/lib/ceph/4709608e-9089-11eb-bb5a-
001a4a000740/osd.6/backup_osd.6
```

6. 以下のように **unit.run** ファイルを編集します。

a. **unit.run** ファイルを確認します。

例

```
[root@host01 osd.3]# cat unit.run

/bin/podman run --rm --ipc=host --authfile=/etc/ceph/podman-auth.json --net=host --
entrypoint /usr/bin/ceph-osd --privileged --group-add=disk --name ceph-4709608e-9089-
11eb-bb5a-001a4a000740-osd.6 -d --log-driver journald --conmon-pidfile /run/ceph-
4709608e-9089-11eb-bb5a-001a4a000740@osd.6.service-pid --cidfile /run/ceph-
4709608e-9089-11eb-bb5a-001a4a000740@osd.6.service-cid -e
CONTAINER_IMAGE=registry.redhat.io/rhceph-alpha/rhceph-5-
rhel8@sha256:9aaea414e2c263216f3cdcb7a096f57c3adf6125ec9f4b0f5f65fa8c43987155
-e NODE_NAME=host01 -v /var/run/ceph/4709608e-9089-11eb-bb5a-
001a4a000740:/var/run/ceph:z -v /var/log/ceph/4709608e-9089-11eb-bb5a-
001a4a000740:/var/log/ceph:z -v /var/lib/ceph/4709608e-9089-11eb-bb5a-
001a4a000740/crash:/var/lib/ceph/crash:z -v /var/lib/ceph/4709608e-9089-11eb-bb5a-
```

```
001a4a000740/osd.6:/var/lib/ceph/osd/ceph-6:z -v /var/lib/ceph/4709608e-9089-11eb-bb5a-001a4a000740/osd.6/config:/etc/ceph/ceph.conf:z -v /dev:/dev -v /run/udev:/run/udev -v /sys:/sys -v /var/lib/ceph/4709608e-9089-11eb-bb5a-001a4a000740/selinux:/sys/fs/selinux:ro -v /run/lvm:/run/lvm -v /run/lock/lvm:/run/lock/lvm registry.redhat.io/rhceph-alpha/rhceph-5-rhel8@sha256:9aaea414e2c263216f3cdcb7a096f57c3adf6125ec9f4b0f5f65fa8c43987155 -n osd.6 -f --setuser ceph --setgroup ceph --default-log-to-file=false --default-log-to-stderr=true --default-log-stderr-prefix="debug"
```

b. ファイルを編集します。

例

```
[root@host01 osd.3]# vi unit.run
```

i. **/bin/podman run --rm --ipc=host --authfile=/etc/ceph/podman-auth.json --net=host --entrypoint /usr/bin/ceph-osd** を /bin/podman run -it --entrypoint /bin/bash に置き換えます。

ii. レジストリーの詳細の後にコンテンツを削除します。例えば、上記のファイルでは、**registry.redhat.io/rhceph-alpha/rhceph-5-rhel8@sha256:9aaea414e2c263216f3cdcb7a096f57c3adf6125ec9f4b0f5f65fa8c43987155** 以降のすべての内容を削除できます。この例では、次のコンテンツが削除されています。

例

```
-n osd.6 -f --setuser ceph --setgroup ceph --default-log-to-file=false --default-log-to-stderr=true --default-log-stderr-prefix="debug"
```

iii. ファイルを保存します。

c. 変更された **unit.run** ファイルをチェックします。

例

```
[root@host01 osd.6]# cat unit.run

/bin/podman run -it --entrypoint /bin/bash --privileged --group-add=disk --name ceph-4709608e-9089-11eb-bb5a-001a4a000740-osd.6 -d --log-driver journald --conmon-pidfile /run/ceph-4709608e-9089-11eb-bb5a-001a4a000740@osd.6.service-pid --cidfile /run/ceph-4709608e-9089-11eb-bb5a-001a4a000740@osd.6.service-cid -e CONTAINER_IMAGE=registry.redhat.io/rhceph-alpha/rhceph-5-rhel8@sha256:9aaea414e2c263216f3cdcb7a096f57c3adf6125ec9f4b0f5f65fa8c43987155 -e NODE_NAME=host01 -v /var/run/ceph/4709608e-9089-11eb-bb5a-001a4a000740:/var/run/ceph:z -v /var/log/ceph/4709608e-9089-11eb-bb5a-001a4a000740:/var/log/ceph:z -v /var/lib/ceph/4709608e-9089-11eb-bb5a-001a4a000740/crash:/var/lib/ceph/crash:z -v /var/lib/ceph/4709608e-9089-11eb-bb5a-001a4a000740/osd.6:/var/lib/ceph/osd/ceph-6:z -v /var/lib/ceph/4709608e-9089-11eb-bb5a-001a4a000740/osd.6/config:/etc/ceph/ceph.conf:z -v /dev:/dev -v /run/udev:/run/udev -v /sys:/sys -v /var/lib/ceph/4709608e-9089-11eb-bb5a-001a4a000740/selinux:/sys/fs/selinux:ro -v /run/lvm:/run/lvm -v /run/lock/lvm:/run/lock/lvm registry.redhat.io/rhceph-alpha/rhceph-5-rhel8@sha256:9aaea414e2c263216f3cdcb7a096f57c3adf6125ec9f4b0f5f65fa8c43987155
```

7. OSD サービスを起動します。

   構文

   ```
   systemctl start SERVICE_ID_OF_OSD
   ```

   例

   ```
   [root@host01 osd.6]# systemctl start ceph-4709608e-9089-11eb-bb5a-001a4a000740@osd.6.service
   ```

8. OSD サービスのステータスを確認します。

   構文

   ```
   systemctl status SERVICE_ID_OF_OSD
   ```

   例

   ```
   [root@host01 osd.6]# systemctl status ceph-4709608e-9089-11eb-bb5a-001a4a000740@osd.6.service

   ● ceph-4709608e-9089-11eb-bb5a-001a4a000740@osd.6.service - Ceph osd.6 for 4709608e-9089-11eb-bb5a-001a4a000740
     Loaded: loaded (/etc/systemd/system/ceph-4709608e-9089-11eb-bb5a-001a4a000740@.service; enabled; vendor preset: disabled)
     Active: active (running) since Mon 2021-04-05 11:05:40 IST; 44s ago
   ```

9. **podman** サービスを実行して OSD の **CONTAINER_ID** を取得します。

   ```
   [root@host01 osd.6]# podman ps -a

   CONTAINER ID  IMAGE
   COMMAND               CREATED         STATUS             PORTS   NAMES
   0ca11f9df3a5  registry.redhat.io/rhceph-alpha/rhceph-5-rhel8@sha256:9aaea414e2c263216f3cdcb7a096f57c3adf6125ec9f4b0f5f65fa8c439871556 minutes ago   Up 6 minutes ago           ceph-4709608e-9089-11eb-bb5a-001a4a000740-osd.6
   ```

10. RocksDB データベースを再度シャード化するには、以下の手順を実施します。

    a. OSD のコンテナーを入力します。

       構文

       ```
       podman exec -it CONTAINER_ID /bin/bash
       ```

       例

       ```
       [root@host01 osd.6]# podman exec -it 0ca11f9df3a5 /bin/bash
       ```

    b. ファイルシステムの整合性を確認するには、**fsck** コマンドを実行します。

構文

```
ceph-bluestore-tool --path /var/lib/ceph/osd/ceph-OSD_ID/ fsck
```

例

```
[root@0ca11f9df3a5 /]# ceph-bluestore-tool --path /var/lib/ceph/osd/ceph-6/ fsck

fsck success
```

c. 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

```
[root@0ca11f9df3a5 /]# 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
```

d. **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) L P" reshard
```

例

```
root@0ca11f9df3a5 /]# ceph-bluestore-tool --path /var/lib/ceph/osd/ceph-6/ --sharding="m(3) p(3,0-12) O(3,0-13) L P" reshard
```

e. 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

```
[root@0ca11f9df3a5 /]# ceph-bluestore-tool --path /var/lib/ceph/osd/ceph-6/ show-sharding

m(3) p(3,0-12) O(3,0-13) L P
```

f. ファイルシステムの整合性を確認するには、**fsck** コマンドを実行します。

構文

```
ceph-bluestore-tool --path /var/lib/ceph/osd/ceph-OSD_ID/ fsck
```

例

```
[root@0ca11f9df3a5 /]# ceph-bluestore-tool --path /var/lib/ceph/osd/ceph-6/ fsck

fsck success
```

11. OSD を再起動するには、まずコンテナーを終了してから、以下のステップを実行します。

a. OSD サービスを停止します。

構文

```
systemctl stop SERVICE_ID_OF_OSD
```

例

```
[root@host01 ~]# systemctl stop ceph-4709608e-9089-11eb-bb5a-
001a4a000740@osd.6.service
```

b. OSD サービスを復元するには、**unit.run** ファイルをバックアップされたファイルに置き換えます。

例

```
[root@host01 osd.6]# cp /var/lib/ceph/4709608e-9089-11eb-bb5a-
001a4a000740/osd.6/backup_osd.6 /var/lib/ceph/4709608e-9089-11eb-bb5a-
001a4a000740/osd.6/unit.run
```

c. OSD サービスを起動します。

構文

```
systemctl start SERVICE_ID_OF_OSD
```

例

```
[root@host01 osd.6]# systemctl start ceph-4709608e-9089-11eb-bb5a-
001a4a000740@osd.6.service
```

検証

- OSD サービスのステータスを確認します。

構文

```
systemctl status SERVICE_ID_OF_OSD
```

-

例

```
[root@host01 osd.6]# systemctl status ceph-4709608e-9089-11eb-bb5a-001a4a000740@osd.6.service

● ceph-4709608e-9089-11eb-bb5a-001a4a000740@osd.6.service - Ceph osd.6 for 4709608e-9089-11eb-bb5a-001a4a000740
   Loaded: loaded (/etc/systemd/system/ceph-4709608e-9089-11eb-bb5a-001a4a000740@.service; enabled; vendor preset: disabled)
   Active: active (running) since Mon 2021-04-05 11:28:21 IST; 34min ago
```

関連情報

- 詳細については、『Red Hat Ceph Storage Installation Guide』を参照してください。

8.7. 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。

8.7.1. 前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。
- BlueStore OSD

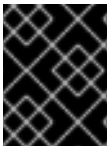
8.7.2. BlueStore 断片化ツールとは

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8.1 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。

8.7.3. 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については、「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。

第9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。

9.1. 前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。

9.2. 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 backend
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。

9.3. 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'"
```

9.4. 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。

9.5. 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
  ```

9.6. 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
  ```

9.7. ダウンロードされたすべてのコンテナーイメージの一覧表示

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一覧表示するには、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/rhel8"
"registry.redhat.io/rhceph-alpha/rhceph-5-
rhel8@sha256:9aaea414e2c263216f3cdcb7a096f57c3adf6125ec9f4b0f5f65fa8c43987155"
```

9.8. 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

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
```

9.9. 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

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
```

9.10. 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
```

9.11. 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 = AQDhcORgW8toCRAAIMzIqWXnh3cGRjqYEa9ikw==
   ```

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-5-rhel8: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10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。

10.1. 前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。

10.2. 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
  ```

  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。

  例

  ```
  2021-06-24T17:51:36.335728+0000 mgr.Ceph5-1.nqikfh [INF] refreshing Ceph5-adm facts
  2021-06-24T17:51:37.170982+0000 mgr.Ceph5-1.nqikfh [INF] deploying 1 monitor(s) instead
  of 2 so monitors may achieve consensus
  2021-06-24T17:51:37.173487+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]
  2021-06-24T17:51:37.174415+0000 mgr.Ceph5-1.nqikfh [INF] Checking pool "nfs-ganesha"
  exists for service nfs.foo
  2021-06-24T17:51:37.176389+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
  2021-06-24T17:51:37.179658+0000 mgr.Ceph5-1.nqikfh [INF] Found osd claims -> {}
  2021-06-24T17:51:37.180116+0000 mgr.Ceph5-1.nqikfh [INF] Found osd claims for
  drivegroup all-available-devices -> {}
  2021-06-24T17:51:37.182138+0000 mgr.Ceph5-1.nqikfh [INF] Applying all-available-devices
  on host Ceph5-adm...
  2021-06-24T17:51:37.182987+0000 mgr.Ceph5-1.nqikfh [INF] Applying all-available-devices
  on host Ceph5-1...
  2021-06-24T17:51:37.183395+0000 mgr.Ceph5-1.nqikfh [INF] Applying all-available-devices
  on host Ceph5-2...
  ```

```
2021-06-24T17:51:43.373570+0000 mgr.Ceph5-1.nqikfh [INF] Reconfiguring node-
exporter.Ceph5-1 (unknown last config time)...
2021-06-24T17:51:43.373840+0000 mgr.Ceph5-1.nqikfh [INF] Reconfiguring daemon node-
exporter.Ceph5-1 on Ceph5-1
```

- 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
  ```

- 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。

10.3. 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お勧めします。

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。

10.4. 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)。

ディスク使用率

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。

10.5. 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。

10.5.1. 前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。

10.5.2. 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。

10.5.3. 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一覧表示するには、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、パッチとアップデート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。