Red Hat Ceph Storage 7

ストレージストラテジーガイド

Red Hat Ceph Storage クラスターのストレージストラテジーの作成

Last Updated: 2024-03-27

Red Hat Ceph Storage 7 ストレージストラテジーガイド

Red Hat Ceph Storage クラスターのストレージストラテジーの作成

法律上の通知

Copyright © 2024 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at
http://creativecommons.org/licenses/by-sa/3.0/
. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux ® is the registered trademark of Linus Torvalds in the United States and other countries.

Java ® is a registered trademark of Oracle and/or its affiliates.

XFS ® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL ® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js ® is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack ® Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

概要

本書では、CRUSH 階層の作成、配置グループの数の見積り、作成するストレージプールのタイプの決定、プールの管理など、ストレージストラテジーの作成方法を説明します。 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、Red Hat CTO である Chris Wright のメッセージ をご覧ください。

目次

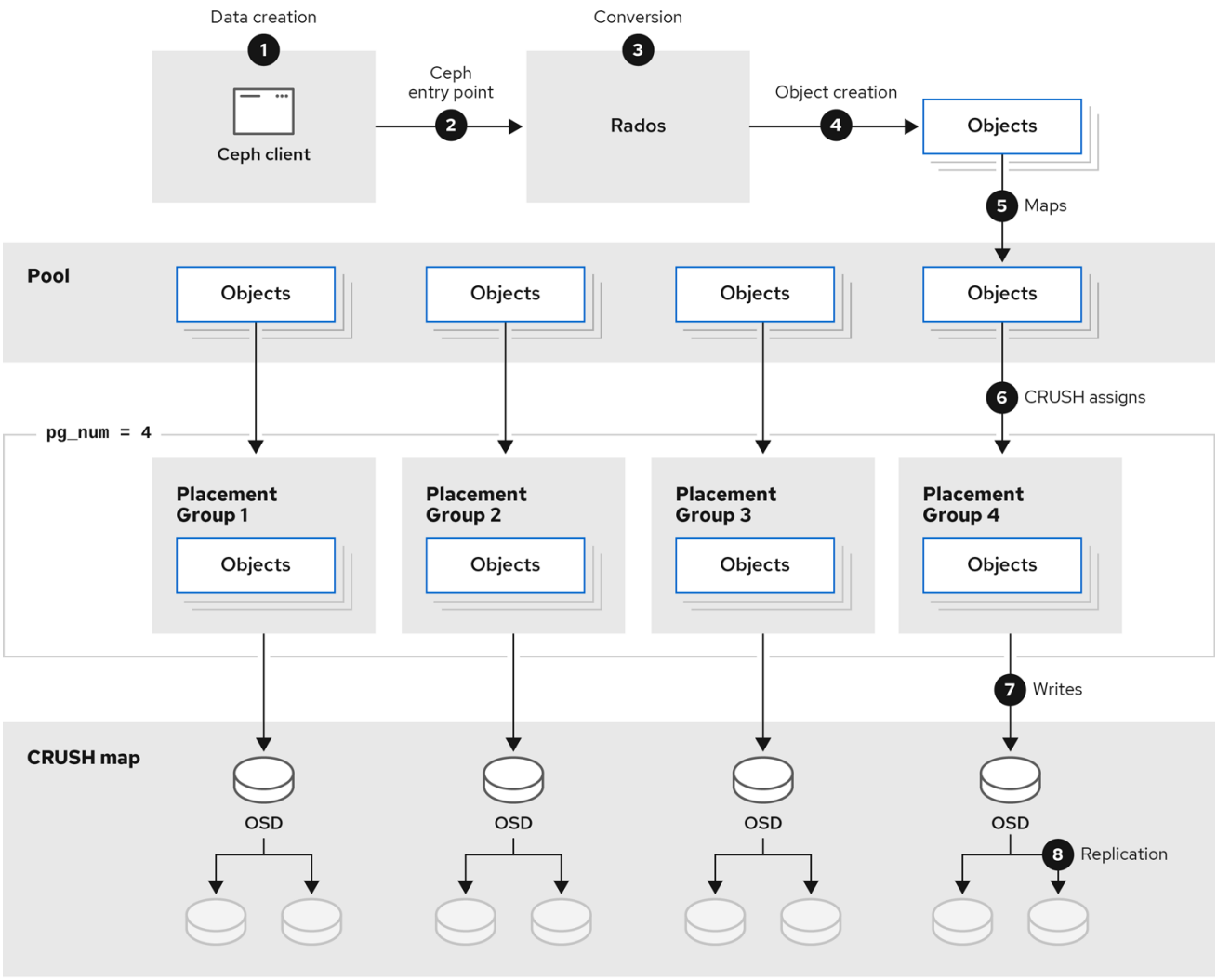
第1章 概要

Ceph クライアントの視点から、Ceph ストレージクラスターとの対話は非常に簡単なものになります。

1. クラスターへの接続
2. プール I/O コンテキストの作成

この非常にシンプルなインターフェイスは、Ceph クライアントが定義するストレージストラテジーのいずれかを選択する方法と同じです。ストレージストラテジーはすべて、ストレージ容量およびパフォーマンスにおいて Ceph クライアントを認識しません。

次の図は、クライアントから Red Hat Ceph Storage クラスターへの論理データフローを示しています。



1.1. ストレージストラテジーの概要

ストレージストラテジーとは、特定のユースケースに対応するためのデータを保管する手法を指します。たとえば、OpenStack のようなクラウドプラットフォーム用のボリュームおよびイメージを格納する必要がある場合は、SSD ベースのジャーナルで、妥当なパフォーマンスの SAS ドライブにデータを保存することを選択できます。これとは対照に、S3 または Swift 準拠のゲートウェイのオブジェクトデータを保存する必要がある場合は、従来の SATA ドライブなどの、より経済的なオプションを選択で

きます。Ceph は、同じ Ceph クラスターで両方のシナリオに対応することができますが、プラットフォーム (OpenStack では SAS/SSD ストレージストラテジーなど) を提供し、オブジェクトストアに SATA ストレージを提供する手段が必要です。

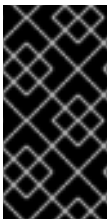
ストレージストラテジーには、ストレージメディア (ハードドライブ、SSD、rest など)、ストレージメディア、配置グループの数、およびプールインターフェイスのパフォーマンスおよび失敗ドメインを設定する CRUSH マップが含まれます。Ceph では、複数のストレージストラテジーがサポートされます。ユースケース、コスト/分散によるパフォーマンスに関するトレードオフおよびデータの持続性は、ストレージ戦略を駆動する主な考慮事項です。

1. **ユースケース:** Ceph は大容量のストレージを提供し、多くのユースケースをサポートします。たとえば、Ceph Block Device クライアントは、OpenStack などのクラウドプラットフォームの主要なストレージバックエンドです。OpenStack の場合、コピーオンライトのクローン作成などの高パフォーマンスな機能と共にボリュームおよびイメージの制限のないストレージを提供。同様に、Ceph は OpenShift 環境にコンテナーベースのストレージを提供できます。これとは対照的に、Ceph Object Gateway クライアントは、音声、ビットマップ、ビデオなどのオブジェクト向けの RESTful S3 準拠のオブジェクトおよび Swift 準拠のオブジェクトストレージを提供するクラウドプラットフォームの主要なストレージバックエンドです。

2. **パフォーマンスのコスト/利点:** 高速さはより優れています。大きい方が優れています。持続性が高くなりました。ただし、ベスト的の質ごとに、対応するコストや利益のトレードオフに価格があります。パフォーマンスの観点からでは、以下のユースケースを考慮してください。SSD は、比較的小規模なデータおよびジャーナリングのために非常に高速ストレージを提供できます。データベースやオブジェクトインデックスを格納すると、非常に高速な SSD のプールから利点がある場合もありますが、他のデータにとっては非常に高価な高価です。SSD ジャーナリングのある SAS ドライブは、ボリュームやイメージを安価かつ高速なパフォーマンスで提供できます。SSD ジャーナリングなしで SATA ドライブを使用すると、パフォーマンスが全体的に低下します。OSD の CRUSH 階層を作成する場合は、ユースケースと許容コスト/パフォーマンスのトレードオフを考慮する必要があります。

3. **耐久性:** 大規模なクラスターでは、ハードウェア障害は予想され、例外ではありません。ただし、データの損失やサービスの中断は、受け入れられなかったままになります。このため、データの持続性は非常に重要です。Ceph は、オブジェクトの複数のディープコピー、またはイレイジャーコーディングおよび複数のコーディングのチャンクでデータの持続性に対応します。複数のコピーまたはコロケーションチャンクが、さらにコストや適切なトレードオフを示します。このトレードオフは、コピーが少なくなる、またはチャンクが少なくなるのが困難です。ただし、動作が低下した状態でサービスの書き込み要求の原因になる可能性があります。通常、2 つの追加のコピー (**size = 3**) または 2 つのコーディングチャンクを持つオブジェクトを使用すると、クラスターが復旧する間にクラスターが動作が低下した状態で書き込みを行うことができます。CRUSH アルゴリズムは、Ceph が、クラスター内の異なる場所に追加のコピーまたはコーディングしたチャンクを保存することでこのプロセスを禁止します。これにより、1 つのストレージデバイスまたはノードに障害が発生しても、データ損失の妨げに必要なコピーやコードチャンクがすべて失われないようにします。

ストレージストラテジーでパフォーマンスのトレードオフやデータの耐性を確保し、それをストレージプールとして Ceph クライアントに提示することができます。



重要

Ceph のオブジェクトのコピーやブロックのチャンクにより RAID が廃止されます。Ceph はすでにデータの持続性に対応しており、質の低い RAID ではパフォーマンスに悪影響があり、RAID を使用してデータを復元すると、ディープコピーや消失訂正を使用するよりもはるかにスピードが遅くなるので、RAID は使用しないでください。

1.2. ストレージストラテジーの設定

ストレージストラテジーの設定は、Ceph OSD を CRUSH 階層に割り当て、プールの配置グループの数を定義し、プールを作成します。一般的な手順は以下のとおりです。

1. **ストレージストラテジーの定義:** ストレージストラテジーでは、ユースケース、コスト/利点のパフォーマンストレードオフおよびデータ永続性を分析する必要があります。次に、そのユースケースに適した OSD を作成します。たとえば、パフォーマンスの高いプール用に SSD 対応 OSD を作成できます。高パフォーマンスのブロックデバイスボリュームおよびイメージ用には SAS ドライブ/SSD のジャーナル OSD、低コストストレージ用の SATA 対応 OSD を作成できます。理想としては、ユースケース向けの各 OSD には同じハードウェア設定を持つ必要があり、これによりパフォーマンスプロファイルの一貫性が保たれます。

2. **CRUSH 階層の定義:** Ceph ルールは、CRUSH 階層にあるノード (通常は **ルート**) を選択し、配置グループおよびそれに含まれるオブジェクトを保存するための適切な OSD を特定します。ストレージストラテジーの CRUSH 階層および CRUSH ルールを作成する必要があります。CRUSH ルール設定では、CRUSH の階層はプールに直接割り当てられます。

3. **配置グループの計算:** Ceph はプールを配置グループにシャード化します。プールの配置グループ数を手動で設定する必要はありません。PG Autoscaler は、同じ CRUSH ルールに対して、複数のプールを割り当てた場合に配置グループの最大数が正常な数に収まるように、適切なプールの配置グループ数を設定します。

4. **プールの作成:** 最終的にプールを作成し、複製されたストレージまたはイレイジャーコーディングされたストレージを使用するかどうかを判別する必要があります。プールの配置グループの数、プールのルール、永続性 (サイズまたは **K+M** コーディングチャンク) を設定する必要があります。

プールはストレージクラスターに対する Ceph クライアントのインターフェイスですが、ストレージストラテジーは Ceph クライアントに対しては完全に透過的です (容量とパフォーマンスを除く)。

第2章 CRUSH 管理の概要

Controlled Replication Under Scalable Hashing (CRUSH) アルゴリズムは、コンピューティングデータストレージの場所によるデータの格納および取得方法を決定します。

	十分な高度な技術は、マジックなものと主導しているものと言えます。	
		-- Arthur C. Clarke

2.1. CRUSH の概要

ストレージクラスターの CRUSH マップは、CRUSH 階層内のデバイスの場所と、Ceph がデータをどのように保管するかを決定する各階層のルールを記述します。

CRUSH マップには、最低でもノードの階層が 1 つ含まれ、残されます。Ceph の buckets という階層のノードは、その種別で定義されるストレージの場所の集約です。たとえば、行、ラック、シャーシ、ホスト、およびデバイスなどがあります。階層の各リーフは、ストレージデバイスのリストにおけるストレージデバイスの 1 つを基本的に設定します。リーフは常に 1 つのノードまたは bucket に含まれます。CRUSH マップには、CRUSH ストアとデータの取得方法を決定するルールのリストもあります。

注記

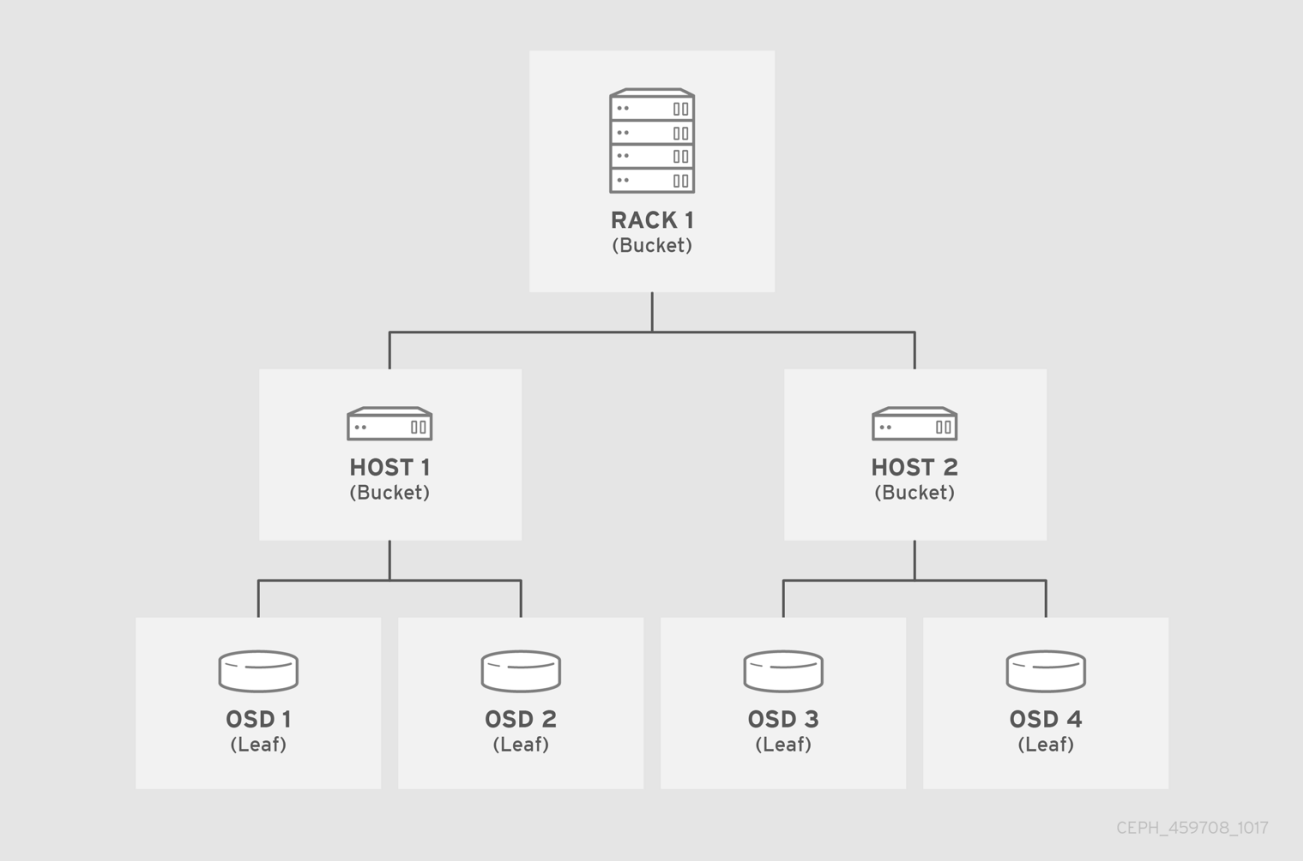
OSD をクラスターに追加する際に、ストレージデバイスが CRUSH マップに追加されます。

CRUSH アルゴリズムは、デバイスごとの加重値に従ってストレージデバイス間でデータオブジェクトを分散します。これは、統一された確率分を分散します。CRUSH は、管理者が定義する階層のクラスターに応じて、オブジェクトとそのレプリカ、または消去したチャンクを分散します。CRUSH マップは、利用可能なストレージデバイスと、ルールにそれらを含む論理バケット、およびルールを使用する各プールで、それぞれを表します。

障害ドメインまたはパフォーマンスドメイン全体で OSD に配置グループをマップするために、CRUSH マップはバケットタイプの階層的なリストを定義します。これは、生成された CRUSH マップの **types** の下にあります。バケット階層を作成する目的は、リーフノードを障害ドメインまたはパフォーマンスドメインか、その両方で分離することです。障害ドメインには、ホスト、シャーシ、ラック、電源分散ユニット、Pod、行、部屋、およびデータセンターが含まれます。パフォーマンスドメインには、特定の設定の障害ドメインおよび OSD が含まれます。たとえば、SSD ジャーナル、SATA ドライブなどを使用した SSD、SAS ドライブなどです。デバイスは **hdd**、**ssd** および **nvme** などの **class** の概念を持ち、デバイスのクラスを使用して CRUSH 階層をより迅速にビルドします。

OSD を表すリーフノードを除き、階層の残りの部分は任意となり、デフォルトタイプが要求に適さない場合、独自のニーズに応じて定義することができます。CRUSH のマップのバケットタイプを組織のハードウェアの命名規則に適合し、物理ハードウェア名を反映するインスタンス名を使用することが推奨されます。命名プラクティスを使用すると、OSD または他のハードウェアの誤作動時に、クラスターの管理と問題のトラブルシューティングを容易にし、管理者がホストまたは他のハードウェアへのリモートアクセスまたは物理的なアクセスが必要な場合に役立ちます。

以下の例では、バケット階層には 4 つのリーフバケット (**osd 1-4**)、ノードバケット (**host 1-2**)、および 1 つのラックノード (**rack 1**) があります。



リーフノードは、CRUSH マップの開始時に **devices** リストに宣言されたストレージデバイスを反映するため、それらをバケットインスタンスとして宣言する必要はありません。階層内の 2 番目のバケットタイプは、通常デバイスを集約し、通常はストレージメディアが含まれるコンピューターで、管理者が好む "node"、"computer"、"server"、"host"、"machine"、"machine" などの用語を使用します。高密度の高い環境では、カードごとおよびシャーシごとに複数のホスト/ノードを確認することが一般的です。カードおよびシャーシの不具合も考慮に入れるようにしてください。たとえば、ノードに障害が発生した場合に、カードやシャーシを引き抜かなければならないようなことになると、非常に多くのホスト/ノードと OSD がダウンすることがあります。

バケットインスタンスを宣言する場合は、その型を指定し、一意の名前を文字列として指定し、負の整数として表現される任意の一意の ID を割り当て、アイテムの合計容量または機能に対する重みを指定し、**straw2** などのバケットアルゴリズムを指定します。また、通常はハッシュアルゴリズム **rjenkins1** を反映した **0** となるハッシュを指定します。バケットには 1 つ以上の項目を指定できます。この項目は、ノードバケットで設定されるか、そのままになります。項目は、項目の相対的な重みを反映する重みを指定できます。

2.1.1. 動的データ配置

Ceph クライアントおよび Ceph OSD はどちらも CRUSH マップと CRUSH アルゴリズムを使用します。

- **Ceph Clients:** CRUSH マップを Ceph クライアントに配布することにより、CRUSH は Ceph クライアントが OSD に直接通信できるようにします。つまり、Ceph クライアントは、単一障害点、パフォーマンスのボトルネック、集中ルックアップサーバーにおける接続制限、ストレージクラスターのスケーラビリティーへの物理的な制限などの集中オブジェクトルックアップテーブルを回避します。

- **Ceph OSD:** CRUSH マップを Ceph OSD に分散することにより、Ceph は OSD を変換してレプリケーション、バックフィル、およびリカバリーを処理します。つまり、Ceph OSD は Ceph クライアントの代わりにオブジェクトレプリカ (または共存するチャンク) のストレージを処理

することを意味します。また、Ceph OSD はクラスター (バックフィル) をクラスターを再分散し、障害から動的に回復するのに十分なクラスターについて把握できることを意味します。

2.1.2. CRUSH 障害ドメイン

複数のオブジェクトレプリカまたは **M** イレイジャーコーディングチャンクを使用するとデータの損失を防ぐことができますが、高可用性に対応するには十分ではありません。Ceph Storage クラスターの基礎となる物理組織を反映することで、CRUSH は、関連するデバイスの障害についてのアドレス指定元的なソースをモデル化できます。クラスターのトポロジーをクラスターマップにエンコードすることで、CRUSH 配置ポリシーは別々の障害ドメインにわたって、オブジェクトレプリカまたはイレイジャーコーディングチャンクを別々に使用できます。また、必要な疑似ランダム分散も依然として維持できます。たとえば、同時障害の可能性に対応するには、異なるシェルフ、ラック、電源、コントローラーまたは物理的な場所を使用するデバイス上または消去するチャンクがデバイスにあることが望ましい場合があります。これは、データ損失を回避し、クラスターが動作が低下した状態で動作できるようにします。

2.1.3. CRUSH パフォーマンスドメイン

CRUSH マップは複数の階層をサポートし、ハードウェアパフォーマンスプロファイルのタイプを別のタイプから分離できます。たとえば、CRUSH はハードディスクドライブと SSD に別の階層を 1 つ作成できます。パフォーマンスドメイン: ベースとなるハードウェアのパフォーマンスプロファイルを把握する階層は、さまざまなパフォーマンス特性をサポートする必要があるため、注目されています。運用上、それらは複数の **root** タイプバケットを持つ CRUSH マップです。ユースケースの例を以下に示します。

- **Object Storage:** S3 および Swift インターフェイスのオブジェクトストレージバックエンドとして機能する Ceph ホストは、仮想マシンに適さない SATA ドライブなど、より安価なストレージメディアを利用する場合があります。また、オブジェクトストレージのギガバイトあたりのコストを軽減しつつ、クラウドプラットフォームでボリュームやイメージを保管するより高性能なストレージホストとより安価なストレージホストを分離することができます。HTTP はオブジェクトストレージシステムでボトルネックとなる傾向があります。

- **コールドストレージ:** コールドストレージ用に設計されたシステム (アクセス頻度が低いデータや、パフォーマンス要件が緩和されたデータの取得) では、より安価なストレージメディアやイレイジャーコーディングを利用できます。ただし、イレイジャーコーディングには、追加の RAM および CPU が必要になることがあり、そのため、オブジェクトストレージまたは仮想マシンに使用されるホストからの RAM および CPU 要件とは異なります。

- **SSD でバックアップされるプール:** SSD は高価ですが、ハードドライブよりも大きな利点があります。SSD にはシーク時間がなく、合計スループットが提供されます。ジャーナリングに SSD を使用することに加え、クラスターは、SSD のバックエンドプールをサポートできます。一般的なユースケースには、高パフォーマンス SSD プールが含まれます。たとえば、Ceph Object Gateway の **.rgw.buckets.index** プールを SATA ドライブではなく SSD にマッピングすることができます。

CRUSH マップは、デバイス **クラス** の概念をサポートします。Ceph はストレージデバイスの要素を検出し、自動的に **hdd**、**ssd**、**nvme** などのクラスを割り当てることができます。ただし、CRUSH はこれらのデフォルトに限定されません。たとえば、CRUSH の階層を使用して、異なるタイプのワークロードを区切られることもできます。たとえば、SSD は、ジャーナルまたはログ先行書き込み、バケットインデックス、または raw オブジェクトストレージに使用される場合があります。CRUSH は **ssd-bucket-index**、**ssd-object-storage** などの異なるデバイスクラスをサポートできるため、Ceph は異なるワークロードに同じストレージメディアを使用しないようにします。これによりパフォーマンスは予測可能で一貫性が高くなります。

バックグラウンドで、Ceph は各デバイスクラスの CRUSH ルートを生成します。これらのルートは、OSD でデバイスクラスを設定または変更することによってのみ変更する必要があります。次のコマンドを使用して、生成されたルートを表示できます。

例

```
[ceph: root@host01 /]# ceph osd crush tree --show-shadow
ID   CLASS  WEIGHT    TYPE NAME
-24    ssd   4.54849  root default~ssd
-19    ssd   0.90970      host ceph01~ssd
  8    ssd   0.90970          osd.8
-20    ssd   0.90970      host ceph02~ssd
  7    ssd   0.90970          osd.7
-21    ssd   0.90970      host ceph03~ssd
  3    ssd   0.90970          osd.3
-22    ssd   0.90970      host ceph04~ssd
  5    ssd   0.90970          osd.5
-23    ssd   0.90970      host ceph05~ssd
  6    ssd   0.90970          osd.6
 -2    hdd  50.94173  root default~hdd
 -4    hdd   7.27739      host ceph01~hdd
 10    hdd   7.27739          osd.10
-12    hdd  14.55478      host ceph02~hdd
  0    hdd   7.27739          osd.0
 12    hdd   7.27739          osd.12
 -6    hdd  14.55478      host ceph03~hdd
  4    hdd   7.27739          osd.4
 11    hdd   7.27739          osd.11
-10    hdd   7.27739      host ceph04~hdd
  1    hdd   7.27739          osd.1
 -8    hdd   7.27739      host ceph05~hdd
  2    hdd   7.27739          osd.2
 -1         55.49022  root default
 -3          8.18709      host ceph01
 10    hdd   7.27739          osd.10
  8    ssd   0.90970          osd.8
-11         15.46448      host ceph02
  0    hdd   7.27739          osd.0
 12    hdd   7.27739          osd.12
  7    ssd   0.90970          osd.7
 -5         15.46448      host ceph03
  4    hdd   7.27739          osd.4
 11    hdd   7.27739          osd.11
  3    ssd   0.90970          osd.3
 -9          8.18709      host ceph04
  1    hdd   7.27739          osd.1
  5    ssd   0.90970          osd.5
 -7          8.18709      host ceph05
  2    hdd   7.27739          osd.2
  6    ssd   0.90970          osd.6
```

2.2. CRUSH 階層

CRUSH マップは転送されたグラフであるため、複数の階層に対応することができます (例: パフォーマンスドメイン)。CRUSH 階層を作成し、変更する方法は Ceph CLI で行いますが、CRUSH マップのコンパイル、編集、再コンパイル、およびアクティブ化することもできます。

Ceph CLI でバケットインスタンスを宣言する場合には、そのタイプを指定して一意の名前 (文字列) を指定する必要があります。Ceph はバケット ID を自動的に割り当て、アルゴリズムを **straw2** に設定し **rjenkins1** を反映してハッシュを **0** に設定し、重みを設定します。コンパイルされていない CRUSH マップを変更する場合は、バケットに負の整数 (任意) として表現される一意の ID を割り当て、アイテムの合計容量/機能に対する重みを指定し、バケットアルゴリズム (通常は **straw2**)、およびハッシュ (通常はハッシュアルゴリズム **rjenkins1** を反映させ **0**)。

バケットには 1 つ以上の項目を指定できます。項目は、ノードバケット (ラック、行、ホストなど)、またはリーフ (OSD ディスクなど) で設定されます。項目は、項目の相対的な重みを反映する重みを指定できます。

コンパイルした CRUSH マップを変更する場合、以下の構文でノードバケットを宣言できます。

```
[bucket-type] [bucket-name] {
    id [a unique negative numeric ID]
    weight [the relative capacity/capability of the item(s)]
    alg [the bucket type: uniform | list | tree | straw2 ]
    hash [the hash type: 0 by default]
    item [item-name] weight [weight]
}
```

たとえば、上図を使用すると、ホストバケットと 1 つのラックバケットを 2 つ定義します。OSD は、ホストバケット内の項目として宣言されます。

```
host node1 {
    id -1
    alg straw2
    hash 0
    item osd.0 weight 1.00
    item osd.1 weight 1.00
}

host node2 {
    id -2
    alg straw2
    hash 0
    item osd.2 weight 1.00
    item osd.3 weight 1.00
}

rack rack1 {
    id -3
    alg straw2
    hash 0
    item node1 weight 2.00
    item node2 weight 2.00
}
```

注記

前の例では、ラックバケットに OSD が含まれていないことに注意してください。低レベルホストバケットが含まれ、項目エントリーでの重みの合計が含まれます。

2.2.1. CRUSH の場所

CRUSH の場所は、CRUSH マップの階層に関して OSD の場所です。コマンドラインインターフェイスで CRUSH の場所を表すと、CRUSH の場所指定子は、OSD の場所を説明する名前/値のペアのリストを取得します。たとえば、OSD が特定の行、ラック、シャーシ、およびホストにあり、**default** の CRUSH ツリーの一部である場合、その CRUSH の場所は以下のように記述できます。

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

注記:

1. キーの順序は問題にはなりません。
2. キー名 (**=** の左) は有効な CRUSH **type** である必要があります。デフォルトでは、これには、**root**、**datacenter**、**room**、**row**、**pod**、**pdu**、**rack**、**chassis**、および **host** が含まれます。CRUSH マップを編集して、ニーズに合わせてタイプを変更できます。
3. すべての buckets/keys を指定する必要はありません。たとえば、デフォルトでは Ceph は **ceph-osd** デーモンの場所を **root=default host={HOSTNAME}** に自動的に設定します (**hostname -s** からの出力に基づいています)。

2.2.2. バケットの追加

CRUSH 階層にバケットインスタンスを追加するには、バケット名とそのタイプを指定します。バケット名は CRUSH マップで一意である必要があります。

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

たとえば、異なるハードウェアパフォーマンスプロファイルなど、複数の階層を使用する場合 (ハードウェアパフォーマンスプロファイルなど)、ハードウェアのタイプまたはユースケースに基づいてバケットの命名を検討してください。

たとえば、ソリッドステートドライブ (**ssd**) の階層、SSD ジャーナルのある SAS ディスクの階層 (**hdd-journal**)、および SATA ドライブ (**hdd**) に別の階層を作成できます。

```
ceph osd crush add-bucket ssd-root root
ceph osd crush add-bucket hdd-journal-root root
ceph osd crush add-bucket hdd-root root
```

Ceph CLI を出力します。

```
added bucket ssd-root type root to crush map
added bucket hdd-journal-root type root to crush map
added bucket hdd-root type root to crush map
```

重要

バケット名におけるコロン (:) の使用はサポートされません。

階層に必要な各バケットタイプのインスタンスを追加します。以下の例は、SSD ホストのラックがある行のバケットと、オブジェクトストレージのホストのラックの追加を示しています。

```
ceph osd crush add-bucket ssd-row1 row
ceph osd crush add-bucket ssd-row1-rack1 rack
ceph osd crush add-bucket ssd-row1-rack1-host1 host
ceph osd crush add-bucket ssd-row1-rack1-host2 host
ceph osd crush add-bucket hdd-row1 row
ceph osd crush add-bucket hdd-row1-rack2 rack
ceph osd crush add-bucket hdd-row1-rack1-host1 host
ceph osd crush add-bucket hdd-row1-rack1-host2 host
ceph osd crush add-bucket hdd-row1-rack1-host3 host
ceph osd crush add-bucket hdd-row1-rack1-host4 host
```

これらの手順を完了したら、ツリーを表示します。

```
ceph osd tree
```

階層はフラットのままである点に注意してください。CRUSH マップに追加した後に、バケットを階層の位置に移動する必要があります。

2.2.3. バケットの移動

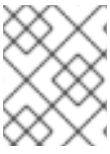
初期クラスターの作成時に、Ceph には **default** という名前のルートバケットのあるの CRUSH マップがあり、初期 OSD ホストは **default** のバケットに表示されます。CRUSH マップにバケットインスタンスを追加する場合、これは CRUSH 階層に表示されますが、必ずしも特定のバケットに表示されるわけではありません。

CRUSH 階層の特定の場所にバケットインスタンスを移動するには、バケット名とそのタイプを指定します。以下に例を示します。

```
ceph osd crush move ssd-row1 root=ssd-root
ceph osd crush move ssd-row1-rack1 row=ssd-row1
ceph osd crush move ssd-row1-rack1-host1 rack=ssd-row1-rack1
ceph osd crush move ssd-row1-rack1-host2 rack=ssd-row1-rack1
```

これらの手順を完了したら、ツリーを表示できます。

```
ceph osd tree
```



注記

ceph osd crush create-or-move を使用して、OSD の移動中に場所を作成することもできます。

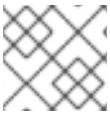
2.2.4. バケットの削除

CRUSH 階層からバケットインスタンスを削除するには、バケット名を指定します。以下に例を示します。

```
ceph osd crush remove {bucket-name}
```

または、以下を実行します。

```
ceph osd crush rm {bucket-name}
```



注記

これを削除するには、バケットを空にする必要があります。

高レベルのバケット (例: **default** などのルート) を削除する場合は、プールがそのバケットを選択する CRUSH ルールを使用するかどうかを確認します。その場合は、CRUSH ルールを変更する必要があります。指定しない場合は、ピアリングに失敗します。

2.2.5. CRUSH バケットのアルゴリズム

Ceph CLI を使用してバケットを作成する場合、Ceph はデフォルトでアルゴリズムを **straw2** に設定します。Ceph は 4 つのバケットアルゴリズムをサポートします。各アルゴリズムは、パフォーマンスと組織の効率間のトレードオフを示しています。使用するバケットタイプが不明な場合は、**straw2** バケットを使用することが推奨されます。バケットアルゴリズムは次のとおりです。

1. **Uniform:** Uniform バケットは、**完全に** 同一の重みを持つデバイスを集約します。たとえば、ハードウェアが送信または廃止されたハードウェアの場合、通常は、同じ物理設定 (一括購入など) を持つ多数のマシンを使用します。ストレージデバイスの重みが完全に一致する場合は、**uniform** されたバケットタイプを使用できます。これにより、CRUSH が一定の時間内にレプリカを統一されたバケットにマップできます。一方向以外の重みでは、別のバケットアルゴリズムを使用する必要があります。

2. **List**: List バケットは、コンテンツをリンクリストとして集約します。RUSH (スケーラブルハッシュでのレプリケーション) $_P$アルゴリズムに基づいて、リストは**拡張クラスター**の自然で直感的な選択です。オブジェクトは適切な確率で最新のデバイスに再配置されるか、以前のように古いデバイスに残ります。アイテムがバケットに追加されると、結果は最適なデータ移行になります。ただし、リストの途中または末尾から削除された項目は、大量の不要な移動が大量に実行され、リストバケットが **縮小されない (またはほとんどない)** 状況に最適です。

3. **Tree**: Tree バケットはバイナリー検索ツリーを使用します。バケットのより大きな項目のセットが含まれる場合、バケットをリスト表示する方が効率的です。RUSH (スケーラブルハッシュでのレプリケーション) $_R$ アルゴリズムに基づいて、ツリーバケットは配置時間を O(log $_n$) に短縮し、はるかに大きなデバイスのセットまたはネストされたバケットの管理に適したものにします。

4. **Straw2 (デフォルト):** List および Tree バケットは、特定の項目に優先順位を指定するか (たとえば、リストの最初の項目を優先するなど)、または項目のサブツリー全体を考慮する必要をなくす方法で分割統治ストラテジーを使用します。レプリカ配置プロセスのパフォーマンスを向上しますが、項目の追加、削除、または再度の重み計算によりバケットの内容が変更される場合に、準最適な再調整的な動作が発生することもあります。**straw2** バケットタイプにより、straw2 の確率とよく似たプロセスで、全アイテムが公平に競争してレプリカ配置を獲得できるようになります。

2.3. CRUSH の CEPH OSD

OSD の CRUSH 階層を作成したら、OSD を CRUSH 階層に追加します。既存の階層から OSD を移動または削除することもできます。Ceph CLI の使用には、以下の値が使用できます。

id

説明

OSD の数値 ID。

型

整数

必須

Yes

例

0

name

説明

OSD のフルネーム。

型

String

必須

Yes

例

osd.0

weight

説明

OSD の CRUSH 加重。

型

Double

必須

Yes

例

2.0

root

説明

OSD が存在する階層またはツリーのルートバケットの名前。

型

キーと値のペア。

必須

Yes

例

root=default、**root=replicated_rule** など

bucket-type

説明

1 つ以上の name-value ペア。ここで、名前はバケットタイプで、値はバケットの名前になります。CRUSH 階層で OSD の CRUSH の場所を指定できます。

型

キーと値のペア。

必須

いいえ

例

datacenter=dc1 room=room1 row=foo rack=bar host=foo-bar-1

2.3.1. CRUSH での OSD の表示

ceph osd crush tree コマンドは、CRUSH バケットと項目をツリービューで出力します。このコマンドを使用して、特定のバケット内の OSD のリストを確認します。**ceph osd tree** のような出力が表示されます。

詳細情報を返すには、以下のコマンドを実行します。

```
# ceph osd crush tree -f json-pretty
```

このコマンドは、以下のような出力を返します。

```
[
  {
    "id": -2,
    "name": "ssd",
    "type": "root",
    "type_id": 10,
    "items": [
      {
        "id": -6,
        "name": "dell-per630-11-ssd",
        "type": "host",
        "type_id": 1,
        "items": [
          {
            "id": 6,
            "name": "osd.6",
            "type": "osd",
            "type_id": 0,
            "crush_weight": 0.099991,
            "depth": 2
          }
        ]
      },
      {
        "id": -7,
        "name": "dell-per630-12-ssd",
        "type": "host",
        "type_id": 1,
        "items": [
          {
            "id": 7,
            "name": "osd.7",
            "type": "osd",
            "type_id": 0,
            "crush_weight": 0.099991,
            "depth": 2
```

```
                }
            ]
        },
        {
            "id": -8,
            "name": "dell-per630-13-ssd",
            "type": "host",
            "type_id": 1,
            "items": [
                {
                    "id": 8,
                    "name": "osd.8",
                    "type": "osd",
                    "type_id": 0,
                    "crush_weight": 0.099991,
                    "depth": 2
                }
            ]
        }
    ]
},
{
    "id": -1,
    "name": "default",
    "type": "root",
    "type_id": 10,
    "items": [
        {
            "id": -3,
            "name": "dell-per630-11",
            "type": "host",
            "type_id": 1,
            "items": [
                {
                    "id": 0,
                    "name": "osd.0",
                    "type": "osd",
                    "type_id": 0,
                    "crush_weight": 0.449997,
                    "depth": 2
                },
                {
                    "id": 3,
                    "name": "osd.3",
                    "type": "osd",
                    "type_id": 0,
                    "crush_weight": 0.289993,
                    "depth": 2
                }
            ]
        },
        {
            "id": -4,
            "name": "dell-per630-12",
            "type": "host",
            "type_id": 1,
```

```
            "items": [
                {
                    "id": 1,
                    "name": "osd.1",
                    "type": "osd",
                    "type_id": 0,
                    "crush_weight": 0.449997,
                    "depth": 2
                },
                {
                    "id": 4,
                    "name": "osd.4",
                    "type": "osd",
                    "type_id": 0,
                    "crush_weight": 0.289993,
                    "depth": 2
                }
            ]
        },
        {
            "id": -5,
            "name": "dell-per630-13",
            "type": "host",
            "type_id": 1,
            "items": [
                {
                    "id": 2,
                    "name": "osd.2",
                    "type": "osd",
                    "type_id": 0,
                    "crush_weight": 0.449997,
                    "depth": 2
                },
                {
                    "id": 5,
                    "name": "osd.5",
                    "type": "osd",
                    "type_id": 0,
                    "crush_weight": 0.289993,
                    "depth": 2
                }
            ]
        }
    ]
  }
]
```


2.3.2. OSD の CRUSH への追加

Ceph OSD を CRUSH 階層に追加することは、OSD を起動する前の最後のステップ (**up** および **in** を編集する) であり、Ceph は配置グループを OSD に割り当てます。

Ceph OSD を準備してから、CRUSH 階層を追加する必要があります。Ceph Orchestrator などのデプロイメントユーティリティーは、この手順を実行できます。たとえば、単一ノードで CephOSD を作成します。

構文

```
ceph orch daemon add osd HOST:_DEVICE_,[DEVICE]
```

CRUSH 階層は概念であるため、**ceph osd crush add** コマンドを使用すると、希望する場所の CRUSH 階層に OSD を追加できます。指定する場所は、実際の場所を反映している **はず** です。少なくとも 1 つのバケットを指定すると、コマンドにより OSD を指定する最も具体的なバケットに配置され、**かつ** そのバケットは指定した他のバケットの下に移動します。

OSD を CRUSH 階層に追加するには、以下を実行します。

構文

```
ceph osd crush add ID_OR_NAME WEIGHT [BUCKET_TYPE=BUCKET_NAME ...]
```

重要

root バケットのみを指定した場合、このコマンドは OSD を直接ルートに割り当てます。ただし、CRUSH ルールは OSD がホストまたはシャーシの内部にあり、ホストまたはシャーシはクラスタートポロジーを反映する他のバケットの内部にある **必要** があります。

以下の例では、**osd.0** を階層に追加します。

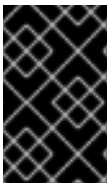
```
ceph osd crush add osd.0 1.0 root=default datacenter=dc1 room=room1 row=foo rack=bar host=foo-bar-1
```

注記

ceph osd crush set または **ceph osd crush create-or-move** を使用して、OSD を CRUSH 階層に追加することもできます。

2.3.3. CRUSH 階層内での OSD の移動

ストレージクラスタートポロジーが変更された場合は、CRUSH 階層内の OSD を移動して、実際の場所を反映させることができます。



重要

CRUSH 階層で OSD を移動すると、Ceph は OSD に割り当てられる配置グループを再計算することを意味します。これにより、データが大幅に再分配される可能性があります。

CRUSH 階層内の OSD を移動するには、以下を実行します。

構文

```
ceph osd crush set ID_OR_NAME WEIGHT root=POOL_NAME
[BUCKET_TYPE=BUCKET_NAME...]
```

注記

ceph osd crush create-or-move を使用して、CRUSH 階層内で OSD を移動することもできます。

2.3.4. CRUSH 階層からの OSD の削除

CRUSH 階層からの OSD 削除は、クラスターから OSD を削除する場合の最初の手順となります。CRUSH マップから OSD を削除すると、CRUSH は配置グループおよびデータリバランスを取得する OSD が再計算されます。詳細は、OSD の追加/削除を参照してください。

実行中のクラスターの CRUSH マップから OSD を削除するには、以下を実行します。

構文

```
ceph osd crush remove NAME
```

2.4. デバイスクラス

Ceph の CRUSH マップは、データの配置を制御するのに余分な柔軟性を提供します。これは、Ceph の最も大きなメリットの 1 つです。初期の Ceph のデプロイメントでは、ハードディスクドライブをほぼ排他的に使用していました。現在、Ceph クラスターは複数のタイプのストレージデバイスで頻繁にビルドされます (HDD、SSD、NVMe、またはさまざまなクラス)。たとえば、クライアントが低速な HDD 上にデータを格納するためのストレージポリシーや、高速 SSD にデータを保存するその他のストレージポリシーを持つように、Ceph Object Gateway デプロイメントにおいて一般的に使用されます。Ceph Object Gateway デプロイメントでは、バケットインデックスの高速 SSD によるプールをサポートする可能性があります。また、OSD ノードにも、CRUSH マップには表示されないジャーナルまたは書き込みログに SSD のみが使用されます。これらの複雑なハードウェアシナリオでは、CRUSH マップを手動で編集する必要がありました。これには、多くの時間と労力が必要となることがありました。ストレージデバイスのクラスごとに、異なる CRUSH 階層を指定する必要はありません。

CRUSH ルールは、CRUSH 階層の用語で機能します。ただし、同じホスト内に異なるストレージデバイスのクラスが存在する場合、このプロセスはより複雑になり、デバイスの各クラスに複数の CRUSH 階層を作成し、CRUSH 階層管理の多くを自動化する **osd crush update on start** オプションを無効にします。デバイスクラスは、使用するデバイスのクラスに対して CRUSH ルールに指示することで、この適合性を排除します。これにより、CRUSH 管理タスクを単純化します。

注記

ceph osd tree コマンドには、デバイスクラスを反映する列があります。

2.4.1. デバイスクラスの設定

OSD にデバイスクラスを設定するには、次のコマンドを実行します。

構文

```
ceph osd crush set-device-class  CLASS OSD_ID [OSD_ID..]
```

例

```
[ceph: root@host01 /]# ceph osd crush set-device-class hdd osd.0 osd.1
[ceph: root@host01 /]# ceph osd crush set-device-class ssd osd.2 osd.3
[ceph: root@host01 /]# ceph osd crush set-device-class bucket-index osd.4
```

注記

Ceph はクラスをデバイスに自動的に割り当てる場合があります。ただし、クラス名は単に任意の文字列です。**hdd**、**ssd**、**nvme** に準拠する必要はありません。前述の例では、**bucket-index** という名前のデバイスクラスが、Ceph Object Gateway プールが排他的バケットインデックスワークロードを使用する SSD デバイスを示す場合があります。すでに設定されているデバイスクラスを変更するには、最初に **ceph osd crush rm-device-class** を使用します。

2.4.2. デバイスクラスの削除

OSD のデバイスクラスを削除するには、以下を実行します。

構文

```
ceph osd crush rm-device-class CLASS OSD_ID [OSD_ID..]
```

例

```
[ceph: root@host01 /]# ceph osd crush rm-device-class hdd osd.0 osd.1
[ceph: root@host01 /]# ceph osd crush rm-device-class ssd osd.2 osd.3
[ceph: root@host01 /]# ceph osd crush rm-device-class bucket-index osd.4
```

2.4.3. デバイスクラスの名前変更

そのクラスを使用するすべての OSD のデバイスクラスの名前を変更するには、以下のコマンドを実行します。

構文

```
ceph osd crush class rename OLD_NAME NEW_NAME
```

例

```
[ceph: root@host01 /]# ceph osd crush class rename hdd sas15k
```

2.4.4. デバイスクラスの一覧表示

CRUSH マップのデバイスクラスをリスト表示するには、以下を実行します。

構文

```
ceph osd crush class ls
```

出力は以下のようになります。

例

```
[
    "hdd",
    "ssd",
    "bucket-index"
]
```

2.4.5. デバイスクラスの OSD の一覧表示

特定のクラスに属するすべての OSD をリスト表示するには、以下のコマンドを実行します。

構文

```
ceph osd crush class ls-osd CLASS
```

例

```
[ceph: root@host01 /]# ceph osd crush class ls-osd hdd
```

出力は、OSD 番号のリストです。以下に例を示します。

```
0
1
2
3
4
5
6
```

2.4.6. クラス別の CRUSH ルールのリスト表示

同じクラスを参照する CRUSH ルールをリスト表示するには、以下のコマンドを実行します。

構文

```
ceph osd crush rule ls-by-class CLASS
```

例

```
[ceph: root@host01 /]# ceph osd crush rule ls-by-class hdd
```

2.5. CRUSH の重み

CRUSH アルゴリズムは、新しいデータオブジェクトを PG に、PG を OSD に割り当てる書き込み要求のための一貫した確率分散で、OSD ごとにテラバイトで重み値を割り当てます (規則により)。このため、ベストプラクティスとして、同じタイプおよびサイズのデバイスとともに CRUSH の階層を作成し、同じ重みを割り当てることが推奨されます。また、パフォーマンス特性がデータの分散に影響を及ぼさなくても、CRUSH 階層にパフォーマンス特性が統一されるように、同じ I/O およびスループット特性でデバイスを使用することが推奨されます。

統一されたハードウェアを使用することは常に実用的な設定ではないため、異なるサイズの OSD デバイスを取り入れ、Ceph が大規模なデバイスにより多くのデータを配信し、小さいデバイスにさらに多くのデータが配信するようにできます。

2.5.1. OSD の CRUSH 重みの設定

CRUSH マップ内の Terabytes で OSD CRUSH 重みを設定するには、以下のコマンドを実行します。

```
ceph osd crush reweight _NAME_ _WEIGHT_
```

ここでは、以下のようになります。

name

説明

OSD のフルネーム。

型

String

必須

Yes

例

osd.0

weight

説明

OSD の CRUSH 加重。これはテラバイト単位で OSD のサイズになります。**1.0** は 1 テラバイトです。

型

Double

必須

Yes

例

2.0

この設定は、OSD を作成するか、OSD の追加直後に CRUSH 重みを調整する際に使用されます。通常、OSD のライフサイクルは変更されません。

2.5.2. バケットの OSD 重みの設定

ceph osd crush reweight を使用すると、時間がかかる可能性があります。以下を実行して、すべての Ceph OSD 加重をバケット (行、ラック、ノードなど) の下で設定できます。

構文

```
osd crush reweight-subtree NAME
```

詳細は以下のようになります。

name は CRUSH バケットの名前です。

2.5.3. OSD の in 重みの設定

ceph osd in および **ceph osd out** の目的上、OSD はクラスター内 (**in**) か、クラスター外 (**out**) のいずれかにあります。これは、モニターする OSD のステータスを記録します。ただし、OSD はクラスター内 **in** となりますが、修正されるまでは依存したくない機能 (ストレージドライブの置き換え、コントローラーの変更など) 生する可能性があります。

以下を実行して (つまり、テラバイト単位でその重みを変更せずに)、以下のコマンドを実行して、特定の OSD の **in** 重みを増減できます。

構文

```
ceph osd reweight ID WEIGHT
```

ここでは、以下のようになります。

- **id** は OSD の番号です。
- **weight** は 0.0 ~ 1.0 の範囲です。 **0** はクラスター内 (**in**) には含まれません (つまり、PG がクラスターに割り当てられていません)。1.0 はクラスター内 (**in**) です (つまり、OSD は他の OSD と同じ数の PG を受信します)。

2.5.4. 使用率による OSD の重みの設定

CRUSH は、新しいデータオブジェクトの PG と PG を OSD に割り当てる書き込み要求のための一貫した確率分散を概観するために設計されています。ただし、クラスターは任意に負荷分散される可能性があります。これは、さまざまな理由で発生する可能性があります。以下に例を示します。

- **複数のプール:** CRUSH 階層に複数のプールを割り当てることができますが、プールには異なる配置グループの数、サイズ (保存するレプリカ数)、およびオブジェクトサイズの特性を持たせることができます。
- **カスタムクライアント:** クライアントからのブロックデバイス、オブジェクトゲートウェイ、ファイルシステムシャードデータなどの Ceph クライアント。統一されたサイズの小さい RADOS オブジェクトとして、データをクラスター全体でオブジェクトとしてストライプ化します。したがって、フォレッシングシナリオを除き、CRUSH は通常、その目的を達成します。ただし、クラスターに不安定な状態が生じるもう 1 つのケースがあります。つまり、**librados** を使用してオブジェクトのサイズを正規化せずにデータを保存することです。このシナリオでは、クラスターがアンバランスになります (例: 100 1 MB と 10 4 MB のオブジェクトを格納すると、他よりも多くのデータを持つ OSD が少なくなります)。
- **可能性:** 統一されたディストリビューションでは、PG が多い OSD と少ない OSD が発生します。OSD が多数あるクラスターの場合、統計外メモリーはさらに省略されます。

以下を実行することで、使用率に従って OSD を再度有効にできます。

構文

```
ceph osd reweight-by-utilization [THRESHOLD_] [WEIGHT_CHANGE_AMOUNT]
[NUMBER_OF_OSDS] [--no-increasing]
```

例

```
[ceph: root@host01 /]# ceph osd test-reweight-by-utilization 110 .5 4 --no-increasing
```

ここでは、以下のようになります。

- **threshold** は、OSD がデータストレージ負荷を高くする使用率が低くなり、割り当てられた PG の数が減ります。デフォルト値は **120** で、120 % を反映しています。 **100** 以降の値はすべて有効なしきい値です。任意です。
- **weight_change_amount** は重みを変更する量です。有効な値は **0.0 - 1.0** より大きいです。デフォルト値は **0.05** です。任意です。
- **number_of_OSDs** は、リライトする OSD の最大数です。大規模なクラスターの場合、OSD の数を reweight に制限すると、影響の大きいリバランスが妨げられます。任意です。
- **no-increasing** は、デフォルトで **off** になっています。**reweight-by-utilization** コマンドまたは **test-reweight-by-utilization** コマンドを使用すると、osd 重みを増やすことができます。このオプションを使用すると、OSD の使用率が低くなっている場合でも、OSD の重みが増加しないようにします。任意です。

重要

大規模なクラスターに **reweight-by-utilization** を実行することが推奨されます。使用率レートは時間の経過と共に変化する場合があります。また、クラスターのサイズやハードウェアの変化により、使用率の変更を反映するために重み付けを更新しなければならない場合があります。使用率の再行を選択した場合には、使用率、ハードウェア、またはクラスターのサイズの変更としてこのコマンドを再実行する必要がある場合があります。

重みを割り当てる上記またはその他の重みのコマンドを実行すると、このコマンドによって割り当てられる重みが上書きされます (例: **osd reweight-by-utilization**、**osd crush weight**、**osd weight**、**in**、または **out**)。

2.5.5. PG ディストリビューションによる OSD の重みの設定

少数の OSD を持つ CRUSH 階層では、一部の OSD が他の OSD よりも長い PG を取得できるため、負荷が高くなることがあります。以下のコマンドを実行して、この状況に対処するために PG ディストリビューションで OSD を再非推奨にすることができます。

構文

```
osd reweight-by-pg POOL_NAME
```

ここでは、以下のようになります。

- **poolname** はプールの名前です。Ceph は、プールの PG を OSD に割り当ててから、このプールの PG ディストリビューションに従って OSD をどのように割り当てるかを検証します。複数のプールを同じ CRUSH 階層に割り当てることができることに注意してください。1 つのプールのディストリビューションに従って OSD を再実行すると、同じ CRUSH 階層に割り当てられた他のプールには、同じサイズ (レプリカの数) と PG が割り当てられていない場合に、意図しない影響が出る可能性があります。

2.5.6. CRUSH ツリーの重みの再計算

CRUSH ツリーバケットは、リーフの重みの合計である必要があります。CRUSH マップの重みを手動で編集する場合は、以下を実行し、CRUSH バケットツリーがバケット内のリーフ OSD の合計を正確に反映するようにする必要があります。

構文

```
osd crush reweight-all
```

2.6. プライマリーアフィニティー

Ceph クライアントがデータの読み取りまたは書き込み時に、適切なセット内のプライマリー OSD を常に問い合わせます。**[2, 3, 4]** セットの場合には、**osd.2** がプライマリーになります。OSD が他の OSD と比較して適していない場合があります (例: ディスクや低速なコントローラーなど)。ハードウェアの使用率を最大限にするために (読み込み操作上) パフォーマンスのボトルネックを防ぐために、CRUSH が OSD を機能セット内のプライマリーセットとして使用することの可能性が低くなるように Ceph OSD のプライマリーアフィニティーを設定できます。

構文

```
ceph osd primary-affinity OSD_ID WEIGHT
```

プライマリーアフィニティーはデフォルトで **1** です (**つまり**、OSD がプライマリーとして機能する可能性があります)。OSD のプライマリー範囲を **0-1** に設定できます。ここで、**0** は OSD をプライマリーとして **使用しない** ことを意味します。**1** は、OSD がプライマリーとして使用される可能性があることを意味します。重みが **<1** の場合は、CRUSH が、プライマリーとして機能する Ceph OSD デーモンを選択する可能性は低くなります。

2.7. CRUSH ルール

CRUSH ルールは、Ceph クライアントがバケットとそれら内の OSD を選択する方法を定義し、プライマリー OSD がバケットとセカンダリー OSD を選択してレプリカやコーディングのチャンクを保存する方法を定義します。たとえば、2 つのオブジェクトレプリカ用に SSD がサポートするターゲット OSD と、3 つのレプリカ用に SAS ドライブがサポートする 3 つのターゲット OSD を選択するルールを作成できます。

ルールは以下の形式を取ります。

```
rule <rulename> {

    id <unique number>
    type [replicated | erasure]
    min_size <min-size>
    max_size <max-size>
    step take <bucket-type> [class <class-name>]
    step [choose|chooseleaf] [firstn|indep] <N> <bucket-type>
    step emit
}
```

id

説明

ルールを識別するための一意の整数。

目的

ルールマスクのコンポーネント。

型

整数

必須

Yes

デフォルト

0

type

説明

レプリケートまたはイレイジャーコーディングされたストレージドライブのルールを説明しています。

目的

ルールマスクのコンポーネント。

型

String

必須

Yes

デフォルト

replicated

有効な値

現在は **replicated** のみ

min_size

説明

プールがこの数よりも小さいレプリカを使用する場合、CRUSH はこのルールを選択しません。

型

整数

目的

ルールマスクのコンポーネント。

必須

Yes

デフォルト

1

max_size

説明

プールがこの数を超えるレプリカを行うと、CRUSH はこのルールを選択しません。

型

整数

目的

ルールマスクのコンポーネント。

必須

Yes

デフォルト

10

step take <bucket-name> [class <class-name>]

説明

バケット名を取り、ツリーを下ってのイテレートを開始します。

目的

ルールのコンポーネント。

必須

Yes

例

step take datastep take data class ssd

step choose firstn <num> type <bucket-type>

説明

特定タイプのバケット数を選択します。通常、この数はプール内のレプリカ数です (プールサイズ)。

- **<num> == 0** の場合は、**pool-num-replicas** バケット (利用可能なすべて) を選択します。
- **<num> > 0 && < pool-num-replicas** の場合は、多くのバケットを選択します。
- **<num> < 0** の場合、これは **pool-num-replicas - {num}** を意味します。

目的

ルールのコンポーネント。

前提条件

step take または **step choose** の後に行います。

例

step choose firstn 1 type row

step chooseleaf firstn <num> type <bucket-type>

説明

{bucket-type} のバケットのセットを選択し、バケットのセットの各バケットのサブツリーからリーフノードを選択します。セットのバケット数は、通常プール内のレプリカ数です (プールサイズ)。

- **<num> == 0** の場合は、**pool-num-replicas** バケット (利用可能なすべて) を選択します。
- **<num> > 0 && < pool-num-replicas** の場合は、多くのバケットを選択します。
- **<num> < 0** の場合、これは **pool-num-replicas - <num>** を意味します。

目的

ルールのコンポーネント。使い方は、2 つの手順を使用してデバイスを選択する必要がなくなります。

前提条件

step take または **step choose** の後に行います。

例

step chooseleaf firstn 0 type row

step emit

説明

現在の値を出力します。また、スタックを除算します。通常、ルールの最後に使用されますが、同じルール内の異なるツリーを選択する際に使用することもできます。

目的

ルールのコンポーネント。

前提条件

step choose の後に行います。

例

step emit

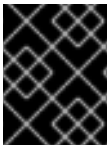
firstn versus indep

説明

CRUSH マップで OSD がダウンする場合に使用する代替ストラテジーを制御します。このルールをレプリケートされたプールで使用する場合はこれを **firstn** にする必要があります。イレイジャーコーディングされたプールの場合は、**indep** にする必要があります。

例

OSD 1、2、3、4、5 に PG が保存されており、3 が落ちています。最初のシナリオでは、**firstn** モードの場合、CRUSH は、計算を調整して 1 および 2 を選択し、次に 3 を選択しますがそれがダウンしていることを検出したため、再試行して 4 と 5 を選択し、新しい OSD 6 を選択します。最終的な CRUSH マッピングの変更は **1、2、3、4、5** から **1、2、4、5、6** になります。2 つ目のシナリオでは、イレイジャーコーディングされたプールに **indep** モードが設定されていると、CRUSH は失敗した OSD 3 の選択を試行し、**1、2、3、4、5** から **1、2、6、4、5**) の最終変換に 6 を選択します。



重要

指定した CRUSH ルールは複数のプールに割り当てることができますが、単一プールで複数の CRUSH ルールを割り当てることはできません。

2.7.1. CRUSH ルールの一覧表示

コマンドラインから CRUSH ルールをリスト表示するには、以下を実行します。

構文

```
ceph osd crush rule list
ceph osd crush rule ls
```

2.7.2. CRUSH ルールのダンプ

特定の CRUSH ルールの内容をダンプするには、以下を実行します。

構文

```
ceph osd crush rule dump NAME
```

2.7.3. CRUSH ルールの追加

CRUSH ルールを追加するには、使用する階層のルートノード、複数の複製するバケットタイプ (例: 'rack'、'row' など)、バケットを選択するモードを指定する必要があります。

構文

```
ceph osd crush rule create-simple RUENAME ROOT BUCKET_NAME FIRSTN_OR_INDEP
```

Ceph は、**chooseleaf** と、指定したタイプのバケットを 1 つ使用してルールを作成します。

例

```
[ceph: root@host01 /]# ceph osd crush rule create-simple deleteme default host firstn
```

以下のルールを作成します。

```
{ "id": 1,
  "rule_name": "deleteme",
  "type": 1,
  "min_size": 1,
  "max_size": 10,
  "steps": [
        { "op": "take",
          "item": -1,
          "item_name": "default"},
        { "op": "chooseleaf_firstn",
          "num": 0,
          "type": "host"},
        { "op": "emit"}]}
```

2.7.4. レプリケートされたプールの CRUSH ルールの作成

レプリケートされたプールに CRUSH ルールを作成するには、以下を実行します。

構文

```
ceph osd crush rule create-replicated NAME ROOT FAILURE_DOMAIN CLASS
```

ここでは、以下のようになります。

- **<name>**: 仮想マシンの名前。
- **<root>**: CRUSH 階層のルート。
- **<failure-domain>**: 障害ドメイン。たとえば、**host** または **rack** です。
- **<class>**: ストレージデバイスクラス。たとえば、**hdd** または **ssd** です。

例

```
[ceph: root@host01 /]# ceph osd crush rule create-replicated fast default host ssd
```

2.7.5. イレイジャーコードプールの CRUSH ルールの作成

イレイジャーコードプールで使用する CRUSH ルールを追加するには、ルール名とイレイジャーコードプロファイルを指定できます。

構文

```
ceph osd crush rule create-erasure RULE_NAME PROFILE_NAME
```

2.7.6. CRUSH ルールの削除

ルールを削除するには、以下を実行し、CRUSH ルール名を指定します。

構文

```
ceph osd crush rule rm NAME
```

2.8. CRUSH の調整可能パラメーターの概要

Ceph プロジェクトでは、多くの変更と新機能が指数関数的に拡張されました。Ceph の最初の商用サポート対象メジャーリリース v0.48 (Argonaut) から始め、Ceph は CRUSH アルゴリズムの特定パラメーターを調整する機能を提供します。つまり、設定はソースコードでフリーズしません。

考慮すべき重要な点を以下に示します。

- CRUSH の値を調整すると、ストレージノード間で一部の PG の変化が発生することがあります。Ceph クラスターが多数のデータを保存する場合には、移動するデータの一部を準備する必要があります。
- **ceph-osd** デーモンおよび **ceph-mon** デーモンは、更新されたマップを受け取るとすぐに、新しい接続の機能ビットを要求するようになります。ただし、すでに接続済みのクライアントはすでに取得され、新機能をサポートしない場合は誤作動します。Ceph クライアントも更新する Ceph Storage Cluster デーモンをアップグレードする場合を確認してください。
- CRUSH の調整可能パラメーターがレガシー以外の値に設定され、後でレガシー値に戻された場合は、その機能をサポートするのに **ceph-osd** デーモンは必要ありません。ただし、OSD ピアリングプロセスでは、古いマップを調べ、理解する必要があります。したがって、クラスターが以前に非レガシー CRUSH 値を使用していた場合は、マップの最新バージョンがレガシーデフォルトの使用に戻されたとしても、古いバージョンの **ceph-osd** デーモンを実行しないでください。

2.8.1. CRUSH のチューニング

CRUSH を調整する前に、すべての Ceph クライアントおよびすべての Ceph デーモンが同じバージョンを使用するようにする必要があります。最近アップグレードした場合は、デーモンを再起動して、クライアントを再接続していることを確認します。

CRUSH パラメーターを調整する最も簡単な方法は、既知のプロファイルに変更します。以下のとおりです。

- **legacy**: v0.47 (pre-Argonaut) 以前のバージョンのレガシー動作。
- **argonaut**: v0.48 (Argonaut) リリースがサポートするレガシーの値。

- **bobtail**: v0.56 (Bobtail) リリースでサポートされる値。
- **firefly**: 0.80 (Firefly) リリースでサポートされる値。
- **hammer**: v0.94 (Hammer) リリースでサポートされる値。
- **jewel**: v10.0.2 (Jewel) リリースでサポートされる値。
- **optimal**: 現在の最適値
- **default**: 新規クラスターの現在のデフォルト値。

実行中のクラスターでプロファイルを選択するには、以下のコマンドを実行します。

構文

```
# ceph osd crush tunables PROFILE
```

注記

これにより、データの移動が生じる場合があります。

通常、アップグレード後に CRUSH パラメーターを設定するか、警告が表示されるようにする必要があります。バージョン v0.74 以降では、CRUSH パラメーターが最適な値に設定されていない場合に、Ceph は健全性についての警告を発行します。最適な値は v0.73 のデフォルトになります。

既存クラスターの調整可能パラメーターを調整すると、警告を削除できます。この結果、データの移動 (10% の可能性) が生じます。これは優先されるルートですが、データの移動がパフォーマンスに影響する可能性があります。以下を使用して、最適なチューニング可能なパラメーターを有効にできます。

```
ceph osd crush tunables optimal
```

パフォーマンスの低下が悪い場合 (たとえば、負荷が非常に多い) か、非常に進捗が行われたか、クライアントの互換性の問題 (カーネルの cephfs または rbd クライアント、または pre-bobtail librados クライアント) がある場合には、以前のプロファイルに戻すことができます。

構文

```
ceph osd crush tunables PROFILE
```

たとえば、pre-v0.48 (Argonaut) 値を復元するには、以下のコマンドを実行します。

例

```
[ceph: root@host01 /]# ceph osd crush tunables legacy
```

2.8.2. CRUSH のチューニング (難しい方法)

すべてのクライアントが最新のコードを実行していることを確認できる場合は、CRUSH マップを抽出して値を変更し、これをクラスターへ再ミラーリングすることで、調整可能パラメーターを調整できます。

- 最新の CRUSH マップを抽出します。

```
ceph osd getcrushmap -o /tmp/crush
```

- 調整可能パラメーターの調整を行います。これらの値は、テストした大規模なクラスターと小規模なクラスターの両方に最適な動作を提供するように見えます。このコマンドが機能するには、**crushtool** に **--enable-unsafe-tunables** 引数も指定する必要があります。このオプションは細心の注意:

```
crushtool -i /tmp/crush --set-choose-local-tries 0 --set-choose-local-fallback-tries 0 --set-choose-total-tries 50 -o /tmp/crush.new
```

- 変更したマップの再インジェクト:

```
ceph osd setcrushmap -i /tmp/crush.new
```

2.8.3. CRUSH のレガシー値

詳細は、CRUSH 調整可能パラメーターのレガシー値を設定できます。

```
crushtool -i /tmp/crush --set-choose-local-tries 2 --set-choose-local-fallback-tries 5 --set-choose-total-tries 19 --set-chooseleaf-descend-once 0 --set-chooseleaf-vary-r 0 -o /tmp/crush.legacy
```

ここでも、特別な **--enable-unsafe-tunables** オプションが必要になります。さらに、上記のように、機能ビットが完全に適用されていないため、レガシー値に戻した後、古いバージョンの **ceph-osd** デーモンを実行する場合は注意が必要です。

2.9. CRUSH マップの編集

通常、Ceph CLI を使用してランタイム時に CRUSH マップを変更すると、CRUSH マップを手動で編集する場合よりも便利です。ただし、デフォルトのバケットタイプの変更や **straw2** 以外のバケットアルゴリズムの使用など、編集を選択できます。

既存の CRUSH マップを編集するには、以下を実行します。

1. CRUSH map の取得。
2. CRUSH マップの逆コンパイル
3. 1 つ以上のデバイス、バケット、およびルールを編集します。
4. CRUSH マップのコンパイル
5. CRUSH マップの設定

特定のプールの CRUSH マップルールを有効にするには、共通ルール番号を特定し、プールの作成時にそのプールのルール番号を指定します。

2.9.1. CRUSH マップの取得

クラスターの CRUSH マップを取得するには、以下を実行します。

構文

```
ceph osd getcrushmap -o COMPILED_CRUSHMAP_FILENAME
```

Ceph は、コンパイルされた CRUSH マップを指定したファイル名に出力 (-o) します。CRUSH マップはコンパイルフォームにあるため、これを編集する前に先にコンパイルする必要があります。

2.9.2. CRUSH マップの逆コンパイル

CRUSH マップをコンパイルするには、以下を実行します。

構文

```
crushtool -d COMPILED_CRUSHMAP_FILENAME -o DECOMPILED_CRUSHMAP_FILENAME
```

Ceph は、コンパイルされた CRUSH マップを逆コンパイル (-d) し、指定したファイル名に出力 (-o) を送信します。

2.9.3. CRUSH マップの設定

クラスターに CRUSH マップを設定するには、以下を実行します。

構文

```
ceph osd setcrushmap -i COMPILED_CRUSHMAP_FILENAME
```

Ceph は、クラスターの CRUSH マップとして指定したファイル名のコンパイル済み CRUSH マップを入力します。

2.9.4. CRUSH マップのコンパイル

CRUSH マップをコンパイルするには、以下を実行します。

構文

```
crushtool -c DECOMPILED_CRUSHMAP_FILENAME -o COMPILED_CRUSHMAP_FILENAME
```

Ceph は、コンパイルされた CRUSH マップを指定したファイル名に保存します。

2.10. CRUSH ストレージストラテジーの例

大規模なハードドライブがサポートするほとんどのプールを OSD に指定するとします (ただし、高速ソリッドステートドライブ (SSD) がサポートする OSD にマッピングされているプールもあります)。CRUSH は、これらのシナリオを容易に処理できます。

デバイスクラスを使用します。プロセスは、各デバイスにクラスを追加するのは簡単です。

構文

```
ceph osd crush set-device-class CLASS OSD_ID [OSD_ID]
```

例

```
[ceph:root@host01 /]# ceph osd crush set-device-class hdd osd.0 osd.1 osd.4 osd.5
[ceph:root@host01 /]# ceph osd crush set-device-class ssd osd.2 osd.3 osd.6 osd.7
```

次に、デバイスを使用するルールを作成します。

構文

```
ceph osd crush rule create-replicated RULENAME ROOT FAILURE_DOMAIN_TYPE DEVICE_CLASS
```

例

```
[ceph:root@host01 /]# ceph osd crush rule create-replicated cold default host hdd
[ceph:root@host01 /]# ceph osd crush rule create-replicated hot default host ssd
```

最後に、ルールを使用するようにプールを設定します。

構文

```
ceph osd pool set POOL_NAME crush_rule RULENAME
```

例

```
[ceph:root@host01 /]# ceph osd pool set cold crush_rule hdd
[ceph:root@host01 /]# ceph osd pool set hot crush_rule ssd
```

1 つの階層が複数のデバイスのクラスに対応できるため、CRUSH マップを手動で編集する必要はありません。

```
device 0 osd.0 class hdd
device 1 osd.1 class hdd
device 2 osd.2 class ssd
device 3 osd.3 class ssd
device 4 osd.4 class hdd
device 5 osd.5 class hdd
device 6 osd.6 class ssd
device 7 osd.7 class ssd

  host ceph-osd-server-1 {
      id -1
      alg straw2
      hash 0
      item osd.0 weight 1.00
      item osd.1 weight 1.00
      item osd.2 weight 1.00
      item osd.3 weight 1.00
  }

  host ceph-osd-server-2 {
      id -2
      alg straw2
      hash 0
      item osd.4 weight 1.00
      item osd.5 weight 1.00
      item osd.6 weight 1.00
      item osd.7 weight 1.00
```

```
}

root default {
    id -3
    alg straw2
    hash 0
    item ceph-osd-server-1 weight 4.00
    item ceph-osd-server-2 weight 4.00
}


rule cold {
    ruleset 0
    type replicated
    min_size 2
    max_size 11
    step take default class hdd
    step chooseleaf firstn 0 type host
    step emit
}


rule hot {
    ruleset 1
    type replicated
    min_size 2
    max_size 11
    step take default class ssd
    step chooseleaf firstn 0 type host
    step emit
}
```

第3章 配置グループ

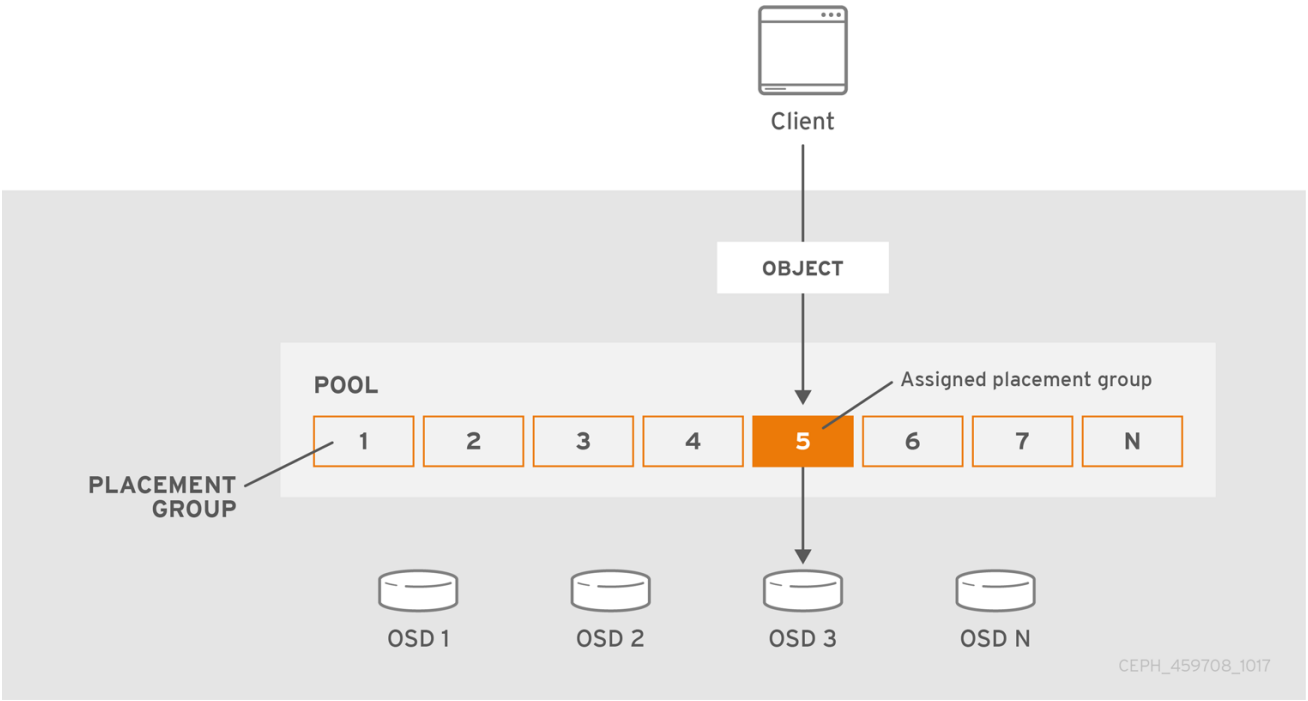
配置グループ (PG) は Ceph クライアントには表示されませんが、Ceph Storage クラスターの重要なロールを果たします。

Ceph Storage Cluster では、ストレージ容量に数万もの OSD が必要になる可能性があります。Ceph クライアントは、オブジェクトをプールに保存します。プールには、クラスターの論理サブセットです。プールに保存されているオブジェクトの数は、数百万以上のものでも簡単に実行できます。オブジェクト数またはそれ以上のシステムが、オブジェクトごとの配置を現実的に追跡できず、適切に実行することができません。Ceph はオブジェクトを配置グループに割り当て、配置グループを OSD に割り当て、動的かつ効率的に分散できるようにします。

	コンピューターのすべての問題は、間接化が過剰に発生する問題を除き、別のレベルの間接方法で解決できます。	
		-- David Wheeler

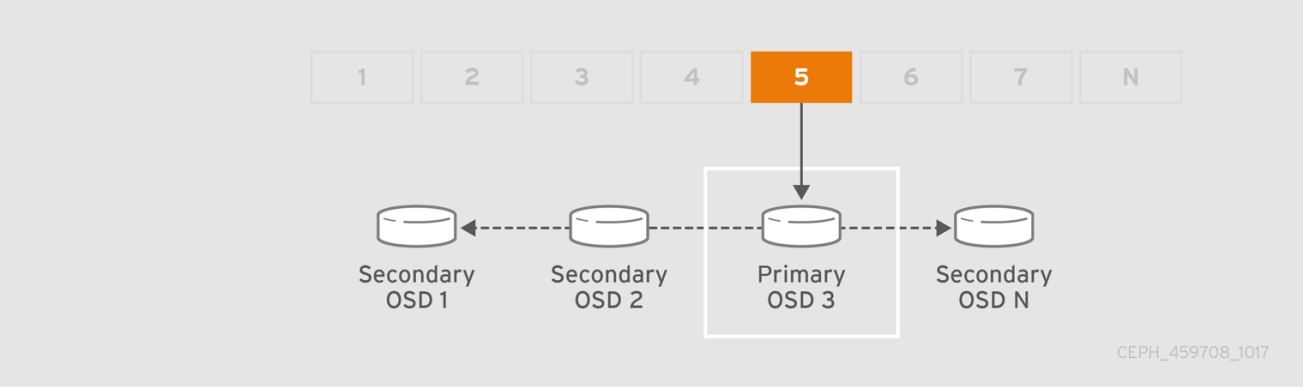
3.1. 配置グループについて

プール内のオブジェクト毎のオブジェクトの配置を追跡することは、スケーリング時に計算が高くなります。スケーリング時に高パフォーマンスを容易にするために、Ceph はプールを配置グループに分割し、各個別のオブジェクトを配置グループに割り当て、配置グループを **プライマリー** OSD に割り当てます。OSD の失敗やクラスターのリバランスを行う場合、Ceph は配置グループ全体を移動するか、複製できます。たとえば、配置グループのすべてのオブジェクトを個別に対応する必要はありません。これにより、Ceph クラスターは効率的にリバランスまたはリカバリーを実行できるようになります。



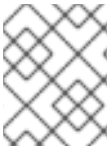
CRUSH が配置グループを OSD に割り当てると、最初の主要な OSD の一連の OSD を計算します。レプリケートされたプールの場合は **osd_pool_default_size** 設定から **1** を引いた、イレイジャーコーディングされたプール用のコーディングチャンク **M** の数が、データを永続的に失わない配置グループを保存できる OSD の数を決定します。プライマリー OSD は CRUSH を使用してセカンダリー OSD を特定し、配置グループのコンテンツをセカンダリー OSD にコピーします。たとえば、CRUSH がオブジェクトを配置グループに割り当て、配置グループがプライマリー OSD として OSD 5 に割り当てられる場合、CRUSH が OSD 1 および OSD 8 がセカンダリー OSD を計算すると、プライマリー OSD 5 は

データを OSD 1 および 8 にコピーします。クライアントの代わりにデータをコピーすると、Ceph はクライアントのインターフェイスを単純化し、クライアントのワークロードを減らします。同じプロセスにより、Ceph クラスターは動的リカバリーおよびリバランスが可能です。



プライマリー OSD が失敗し、クラスターの印が付けられた場合、CRUSH は配置グループを別の OSD に割り当てます。これにより、配置グループ内のオブジェクトのコピーを受け取ります。**Up Set** 内の別の OSD は、プライマリー OSD ロールを想定します。

オブジェクトレプリカ数を増やすか、共存する場合、CRUSH は各配置グループを必要に応じて追加の OSD に割り当てます。



注記

PG は OSD を所有しません。CRUSH は、多くの配置グループを各 OSD 擬似リソースに割り当て、データをクラスター全体で均等に分散できるようにします。

3.2. 配置グループの状態

ceph -s または **ceph -w** コマンドを使用してストレージクラスターのステータスを確認すると、Ceph は配置グループ (PG) のステータスを報告します。PG には 1 つ以上の状態があります。PG マップの PG の最適な状態は **active + clean** 状態です。

activating

PG はピアリングされますが、まだアクティブではありません。

active

Ceph は PG への要求を処理します。

backfill_toofull

宛先 OSD がバックフィル比率を超えているため、バックフィル操作は待機します。

backfill_unfound

不明なオブジェクトが原因でバックフィルが停止しました。

backfill_wait

PG はバックフィルを開始するために行に待機中です。

backfilling

Ceph は、最近の操作のログからコンテンツを同期する必要があるコンテンツを推測する代わりに、PG のコンテンツ全体をスキャンして同期しています。バックフィルは、リカバリーの特別なケースです。

clean

Ceph は PG 内のすべてのオブジェクトを正確に複製します。

作成

Ceph はまだ PG を作成します。

deep

Ceph は、保存されたチェックサムに対して PG データをチェックします。

degraded

Ceph はまだ PG 内の一部のオブジェクトを複製していません。

down

必要なデータを持つレプリカがダウンしているので、PG はオフラインになります。**min_size** レプリカ未満の PG は down とマークされます。バッキング OSD の状態を理解するには、**ceph health detail** を使用します。

forced_backfill

ユーザーが PG を強制する高バックフィルの優先度。

forced_recovery

ユーザーが PG のリカバリーの優先度が高くなる。

incomplete

Ceph は、PG が発生した可能性のある書き込みに関する情報がないか、また正常なコピーを持たないことを検出します。この状態が表示される場合には、必要な情報が含まれる可能性のある失敗した OSD を起動してみてください。イレイジャーコード化されたプールの場合、**min_size** を一時的に減らすと、回復できる可能性があります。

inconsistent

Ceph は、オブジェクトの形式が間違っているサイズなど、PG のオブジェクトの 1 つ以上のレプリカで不整合を検出すると、リカバリーの終了後に 1 つのレプリカにオブジェクトが欠落します。

peering

PG はピアリングプロセスを実行しています。ピアリングプロセスはそれほど遅れることなくクリアされるはずですが、それが維持され、ピアリング状態の PG の数が減らない場合は、ピアリングがスタックしている可能性があります。

peered

PG はピアリングしましたが、プールの設定済み **min_size** パラメーターに到達するのに十分なコピーがないため、クライアント IO にサービスを提供できません。この状態でリカバリーが行われる可能性があるため、PG は最終的に **min_size** まで修復される可能性があります。

recovering

Ceph はオブジェクトとそのレプリカを移行または同期しています。

recovery_toofull

宛先 OSD が 完全な比率 を超えているため、リカバリー操作は待機します。

recovery_unfound

オブジェクトが見つからないため、リカバリーが停止しました。

recovery_wait

PG は、リカバリーを開始するためにインラインで待機中です。

remapped

PG は、指定される CRUSH から異なる OSD のセットに一時的にマッピングされます。

repair

Ceph は PG を確認し、可能であれば見つかった不整合を修復します。

replay

PG は、OSD のクラッシュ後にクライアントが操作を再生するのを待機中です。

snaptrim

snaps のトリミング。

snaptrim_error

snaps のトリムが停止したエラー。

snaptrim_wait

snaps をトリミングするようキューに置かれます。

scrubbing

Ceph は、不整合について PG メタデータを確認します。

分割

Ceph は PG を複数の PG に分割します。

stale

PG は不明な状態です。PG マッピングが変更されてから、モニターは更新を受信していません。

undersized

PG は、設定されたプールレプリケーションレベルよりもコピーが少なくなります。

unknown

ceph-mgr は、Ceph Manager の起動以降 OSD から PG の状態についての情報をまだ受信していません。

関連情報

- 詳細は、ナレッジベースの **Ceph クラスターで可能な配置グループの状態とは** を参照してください。

3.3. 配置グループのトレードオフ

より多くの配置グループに対するすべての OSD 呼び出し間のデータの持続性とデータの分散性以外にも、CPU とメモリーリソースを節約するための最大パフォーマンスを最大化するために最低限必要な数を減らす必要があります。

3.3.1. データの持続性

Ceph はデータの永続的な損失を防ぐように努めます。ただし、OSD が失敗すると、含まれるデータが完全に回復するまで、永続的なデータの損失のリスクが高まります。まれに、永続的なデータ損失を使用することは可能です。以下のシナリオとして、Ceph が 1 つの配置グループのデータを永続的に失った方法が、3 つのデータのコピーで永久に失われるシナリオを説明します。

- OSD が失敗し、これに含まれるオブジェクトのすべてのコピーが失われます。OSD に保管されている配置グループ内のすべてのオブジェクトについて、レプリカ数を 3 から 2 に破棄します。
- Ceph は、新規 OSD を選択して、各配置グループの全オブジェクトの 3 つ目のコピーを再作成して、失敗した OSD に保管されている各配置グループのリカバリーを開始します。
- 新しい OSD が完全にコピー 3 つになるまで、同じ配置グループのコピーを含む 2 つ目の OSD は失敗します。一部のオブジェクトには、コピーが 1 つだけ含まれます。
- Ceph はまだ別の OSD を選択し、オブジェクトをコピーして必要なコピー数を復元します。

- リカバリーが完了するまで、同じ配置グループのコピーを含む 3 つ目の OSD は失敗します。この OSD にオブジェクトの残りのコピーのみが含まれる場合、オブジェクトは永続的に失われます。

ハードウェア障害は例外ではありませんが、予想されます。前述のシナリオを防止するには、リカバリープロセスを最速に行いておく必要があります。クラスターのサイズ、ハードウェア設定、および配置グループの数は、復旧時間の合計における重要なロールを果たします。

小規模なクラスターはすぐにリカバリーしません。

3 つのレプリカプールに 512 の配置グループと共に 10 OSD が含まれるクラスターでは、CRUSH ごとに配置グループ 3 つが提供されます。各 OSD は、ホスト **(512 * 3)** / **10 = ~150** の配置グループになります。最初の OSD が失敗すると、クラスターは 150 のすべての配置グループのリカバリーを同時に開始します。

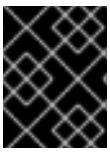
Ceph は、9 つの残りの OSD にわたり、残りの 150 の配置グループをランダムに保存している可能性があります。したがって、残りの OSD は、現在割り当てられている 150 個の配置グループの一部を担当することになるため、残りの各 OSD は、他のすべての OSD にオブジェクトのコピーを送信する可能性が高く、また、いくつかの新しいオブジェクトを受け取る可能性があります。

リカバリー合計時間は、プールをサポートするハードウェアによって異なります。たとえば、10 OSD クラスターにおいて、ホストに 1 TB SSD を持つ OSD が 1 つあり、10 GB/s スイッチが 10 個の各ホストを接続すると、復旧に **M** 分かかります。一方、ホストに 2 つの SATA OSD が含まれ、1 GB/s スイッチが 5 台に接続すると、復元が大幅に長くなります。同様に、このサイズのクラスターで、配置グループの数に直面的には、データの持続性に影響を与えることはありません。配置グループ数は 128 または 8192 で、復元速度は遅くなったり、速くなったりしません。

ただし、同じ Ceph クラスターを 10 OSD ではなく 20 個の OSD に拡張すると、復元を迅速化するため、データの持続性が大幅に向上します。なぜですか ?各 OSD は 150 ではなく、75 の配置グループしか参加しません。20 個の OSD クラスターでは、回復するために同じコピー操作を実行するには、残り 19 個すべての OSD が必要になります。10 OSD クラスターでは、各 OSD は約 100 GB をコピーする必要がありました。各 OSD はそれぞれ 20 個の OSD クラスターでは、それぞれ 50 GB のみをコピーする必要があります。ネットワークがボトルネックである場合、リカバリーは高速になります。つまり、OSD の数が増えると、復旧時間が短縮されます。

大規模なクラスターでは、PG 数は重要です。

exemplary クラスターが 40 OSD に拡大すると、各 OSD は 35 の配置グループのみをホストします。OSD が停止した場合、別のボトルネック解決しないと、復旧時間が減少します。ただし、このクラスターが 200 個の OSD を拡張する場合、各 OSD は約 7 の配置グループのみをホストします。OSD が停止した場合、これらの配置グループにある最大 21 **(7 * 3)** OSD の間に復元が行われます。**復元には、40 の OSD があった場合よりも時間がかかります。つまり、配置グループの数を増やす必要があります。**



重要

リカバリー時間は短い方法に関係ありません。復元中、他の OSD が配置グループの保存時に失敗する可能性があります。

上記の 10 OSD クラスターでは、いずれかの OSD に障害が発生した場合は、約 8 個の配置グループ (つまり、**75 pgs** / **9 osds** が復元されます) には、残りのコピーが 1 つしかありません。残りの 8 つの OSD のいずれかが失敗すると、1 つの配置グループの最後のオブジェクトが失われる可能性があります (つまり、残りの 1 つのコピーのみが復元される **8 pgs** / **8 osds** など)。このため、大規模なクラスターから始まります (例: 50 OSD)。

クラスターのサイズが 20 個の OSD に増加する場合、3 つの OSD ドロップによって破損した配置グループの数が増えます。2 つ目に失われた OSD は、8 ではなく約 2 (つまり **35 pgs** / **19 osds** が復元) に低下し、3 番目に失われた OSD は、残りのコピーを含む 2 つの OSD の 1 つである場合にのみデータ

を失います。つまり、回復時間枠内で 1 つの OSD が失われる確率が **0.0001%** の場合は、10 OSD のクラスターの **8 * 0.0001%** から 20 OSD のクラスターの **2 * 0.0001%** になります。データの耐性が懸念されるため、50 OSD 未満のクラスターで 512 または 4096 の配置グループがほぼ同等になります。

ヒント

つまり、OSD が多いほどリカバリーが速くなり、カスケーディングが配置グループとそのオブジェクトの永続的な損失が発生するリスクが低くなります。

OSD をクラスターに追加する場合、新しい OSD に配置グループおよびオブジェクトが追加されるまでに長い時間がかかる可能性があります。ただし、オブジェクトの低下はなく、OSD を追加するとデータの持続性は影響を受けません。

3.3.2. データディストリビューション

Ceph はホットスポットを避けるようにします。一部の OSD は、その他の OSD よりも多くのトラフィックを受信します。理想的には、CRUSH は配置グループにオブジェクトを均等に割り当て、配置グループが OSD (またはランダムに擬似) に割り当てられる場合でも、プライマリー OSD は、クラスター全体に均等に分散され、ホットスポットやネットワークオーバーサブスクリプションの問題が発生しないようにオブジェクトをストアします。

CRUSH は各オブジェクトの配置グループを計算するが、実際にはこの配置グループ内の各 OSD に保管されるデータの量を認識しないため、**配置グループの数と OSD の数の比率が、データの分散に大きく影響する可能性があります**。

たとえば、3 つのレプリカプールに OSD が 10 個しかない配置グループが 1 つしかない場合、Ceph は CRUSH には他の選択がないために 3 つの OSD だけを使用します。より多くの配置グループを利用できる場合、CRUSH はオブジェクトを OSD 全体に均等に分散させる可能性が高くなります。また、CRUSH は配置グループを OSD に均等に割り当てます。

OSD よりも多くの配置グループの順序が 1 つまたは 2 つまたは 2 つある限り、ディストリビューションも OSD よりも多くの配置グループで設定される必要があります。たとえば、OSD 3 つは 256 の配置グループ、10 個は OSD 用 512 または 1024 の配置グループなどとなります。

OSD と配置グループの割合は、通常、オブジェクトストライピングのような高度な機能を実装する Ceph クライアントのデータ分散の問題を解決します。たとえば、4 TB ブロックデバイスでは、4 MB オブジェクトになる可能性があります。

CRUSH はオブジェクトサイズを考慮しないため、OSD と配置グループ間の比率は、他のケースで不均等なデータ分散に対応しません。librados インターフェイスを使用して、いくつかの比較的小さなオブジェクトといくつかの非常に大きなオブジェクトを格納すると、データの分散が不均一になる可能性があります。たとえば、10 個の OSD 上に 1,000 個の配置グループの中に、合計 100 万個の 4K オブジェクト (合計で 4 GB) が均等に配置されています。各 OSD で **4 GB / 10 = 400 MB** が使用されます。プールに 400 MB オブジェクト 1 つが追加されると、オブジェクトの配置先の配置グループをサポートする 3 つの OSD で **400 MB + 400 MB = 800 MB** が使用され、残りの 7 つの OSD は 400 MB のみで占有されます。

3.3.3. リソースの使用状況

各配置グループ、OSD および Ceph モニターにはメモリー、ネットワーク、および CPU を常時必要とし、復旧中にさらに必要です。配置グループ内のクラスタリングオブジェクトによるこのオーバーヘッドの共有は、主な配置グループのいずれかです。

配置グループの数を最小限にすると、リソースの量が大きくなります。

3.4. 配置グループの数

プール内の配置グループ数では、クラスターのピアがデータおよびリバランスの分散方法などの重要なロールを果たします。小規模なクラスターでは、配置グループの数を増やすことで、大規模なクラスターと比較してパフォーマンスが向上されません。ただし、同じ OSD に多数のプールを持つクラスターは、Ceph OSD がリソースを効率的に使用するように PG 数を考慮しないといけない場合があります。

ヒント

Red Hat は、OSD あたり 100 から 200 PG を推奨します。

3.4.1. 配置グループ計算ツール

配置グループ (PG) 計算ツールは、配置グループの数を計算し、特定のユースケースに対応します。PG の計算ツールは、通常同じルール (CRUSH 階層) を使用して Ceph Object Gateway などの Ceph クライアントを使用する場合に特に役立ちます。小規模なクラスターの配置グループ数 および 配置グループ数の計算 のガイドラインを使用して、PG を手動で計算することもできます。ただし、PG の計算ツールは、PG を計算する方法として推奨されています。

詳細は、Red Hat カスタマーポータル の Ceph Placement Groups (PGs) per Pool Calculator を参照してください。

3.4.2. デフォルトの配置グループ数の設定

プールの作成時に、プールに配置グループの数も作成します。配置グループの数を指定しない場合、Ceph はデフォルト値 **8** を使用します。これは許容範囲を超えるほど低い値です。プールの配置グループ数を増やすことはできますが、妥当なデフォルト値を設定することを推奨します。

```
osd pool default pg num = 100
osd pool default pgp num = 100
```

配置グループ (合計) 数と、オブジェクトに使用される配置グループの数 (PG 設定で使用) の両方を設定する必要があります。これらは等しい必要があります。

3.4.3. 小規模なクラスターの配置グループ数

小規模なクラスターでは、多くの配置グループには利点はありません。OSD の数が増えると、**pg_num** および **pgp_num** が正しい値を選択することが重要になります。これは、クラスターの動作に大きな影響を与えるだけでなく、異常が発生した時のデータの耐久性 (致命的なイベントによってデータ損失が発生する可能性) があるためです。小規模なクラスターでは、PG 計算ツール を使用することが重要です。

3.4.4. 配置グループ数の計算

OSD が 50 を超える場合は、リソース使用状況、データ耐性、分散のバランスを取るために、OSD ごとに約 50 - 100 個の配置グループを推奨します。OSD が 50 未満の場合は、Small クラスターの PG 数などを選択することが理想的です。オブジェクトのプール 1 つに対して、以下の式を使用してベースラインを取得できます。

```
              (OSDs * 100)
Total PGs =  ------------
               pool size
```

プールサイズ は、(**ceph osd erasure-code-profile get** によって返される) レプリケートされたプールのレプリカ数、または異例ジャーコードされたプールの **K+M** 合計になります。

次に、データの永続性を最大化し、データ分散を最大化し、リソースの使用を最小限に抑えるために、Ceph クラスターが設計した内容が適切かどうかを確認する必要があります。

結果は、**最も近い 2 の累乗に切り上げられる必要があります**。数の丸めはオプションですが、CRUSH では配置グループ間でオブジェクト数を均等に分散させることが推奨されます。

OSD が 200 でプールサイズ 3 つのレプリカのクラスターの場合、以下のように PG 数を見積もります。

```
(200 * 100)
----------- = 6667. Nearest power of 2: 8192
     3
```

8192 個の配置グループを 200 個の OSD に分散することで、OSD ごとに約 41 の配置グループを評価します。また、クラスターで使用される可能性のあるプールの数も考慮する必要があります。これは、各プールで配置グループが作成されるので、クラスターで使用される可能性も検討する必要があります。配置グループの最大数 が妥当であることを確認してください。

3.4.5. 配置グループの最大数

オブジェクト格納に複数のデータプールを使用する場合は、プールごとの配置グループの数と、OSD ごとの配置グループの合計数が妥当な数になるように、配置グループの合計数のバランスを取る必要があります。この目的は、システムリソースに負荷をかけたり、ピアリングプロセスを遅らせずに、OSD ごとのさまざまな負荷を達成することにあります。

たとえば、10 個の OSD 上の 512 の配置グループを持つ、10 個のプールで設定される Ceph Storage クラスターでは、10 個を超える OSD に広がる 5120 の配置グループや、OSD ごとに 512 の配置グループがあります。ハードウェアの設定によっては、リソースが多すぎる可能性があります。これとは対照的に、512 個の配置グループをそれぞれに持つ 1,000 プールを作成する場合、OSD はそれぞれの配置グループ 50,000 までを処理し、より多くのリソースが必要になります。OSD ごとの配置グループが多すぎると、特にリバランスまたは復旧時にパフォーマンスが大幅に低下します。

Ceph Storage Cluster には、OSD ごとに最大 300 の配置グループの最大値があります。Ceph 設定ファイルに異なる最大値を設定することができます。

```
mon pg warn max per osd
```

ヒント

Ceph Object Gateway は 10-15 プールでデプロイするので、妥当な最大数に達するにあたり OSD ごとに 100 未満の PG 未満を使用してください。

3.5. 配置グループの自動スケーリング

プール内の配置グループ (PG) 数では、クラスターのピアがデータおよびリバランスの分散方法などの重要なロールを果たします。

PG 数の自動スケーリングにより、クラスターの管理が容易になります。**pg-autoscaling** コマンドは、PG のスケーリングの推奨事項を示します。または、クラスターの使用状況に応じて PG を自動的にスケーリングします。

- 自動スケーリングの動作に関する詳細は、「配置グループの自動スケーリング」 を参照してください。
- 自動スケーリングを有効または無効にする場合は、「配置グループの自動スケーリングモードの設定」 を参照してください。
- 配置グループのスケーリングの推奨事項を表示するには、「配置グループのスケーリングの推奨事項」 を参照してください。
- 配置グループの自動スケーリングを設定するには、「配置グループの自動スケーリングの設定」 を参照してください。
- オートスケーラーをグローバル更新するには、「**noautoscale** フラグの更新」 を参照してください。
- ターゲットプールサイズを設定するには、「ターゲットプールサイズの指定」 を参照してください。

3.5.1. 配置グループの自動スケーリング

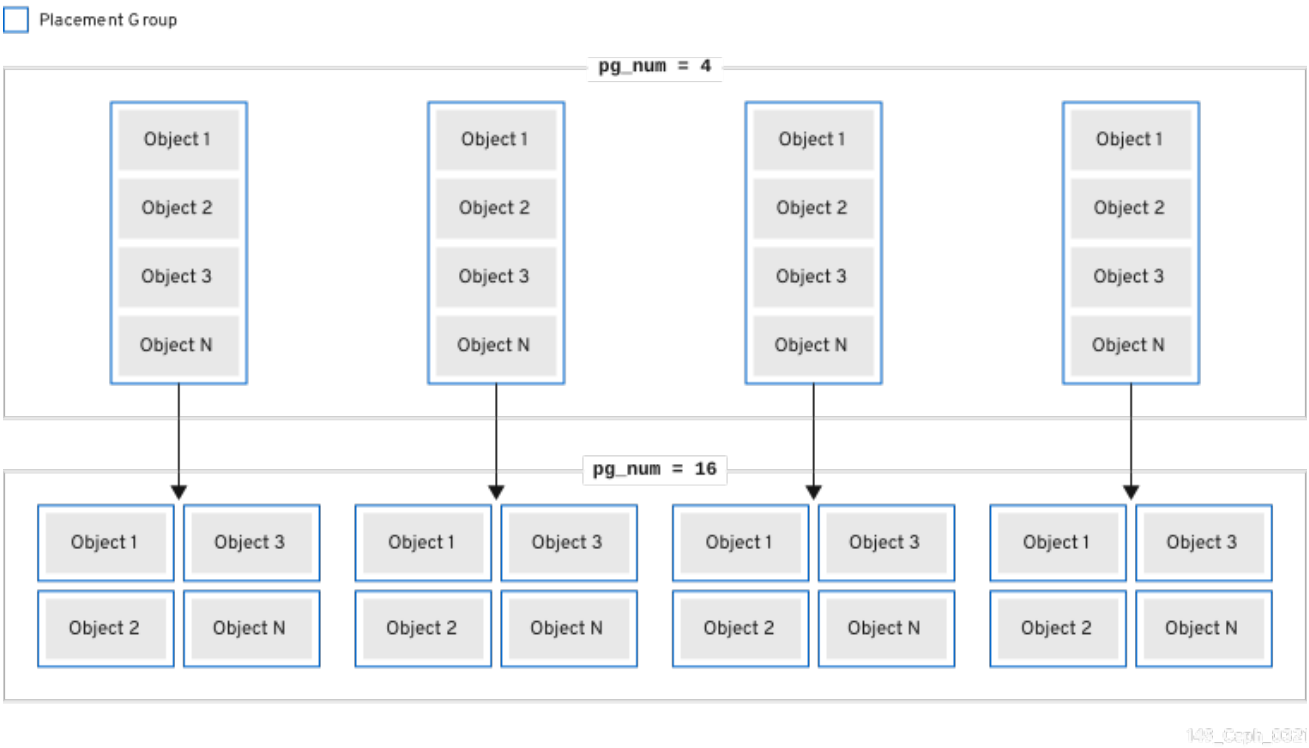
auto-scaler の仕組み

auto-scaler はプールを分析し、サブツリーごとに調整します。各プールは異なる CRUSH ルールにマップされ、各ルールは異なるデバイスにデータを分散できるため、Ceph は階層の各サブツリーを独立して使用することを考慮します。たとえば、クラス **ssd** の OSD にマップするプールと **hdd** クラスの OSD にマップするプールは、それぞれがそれぞれのデバイスタイプの数に依存する最適な PG 数を持ちます。

3.5.2. 配置グループの分割とマージ

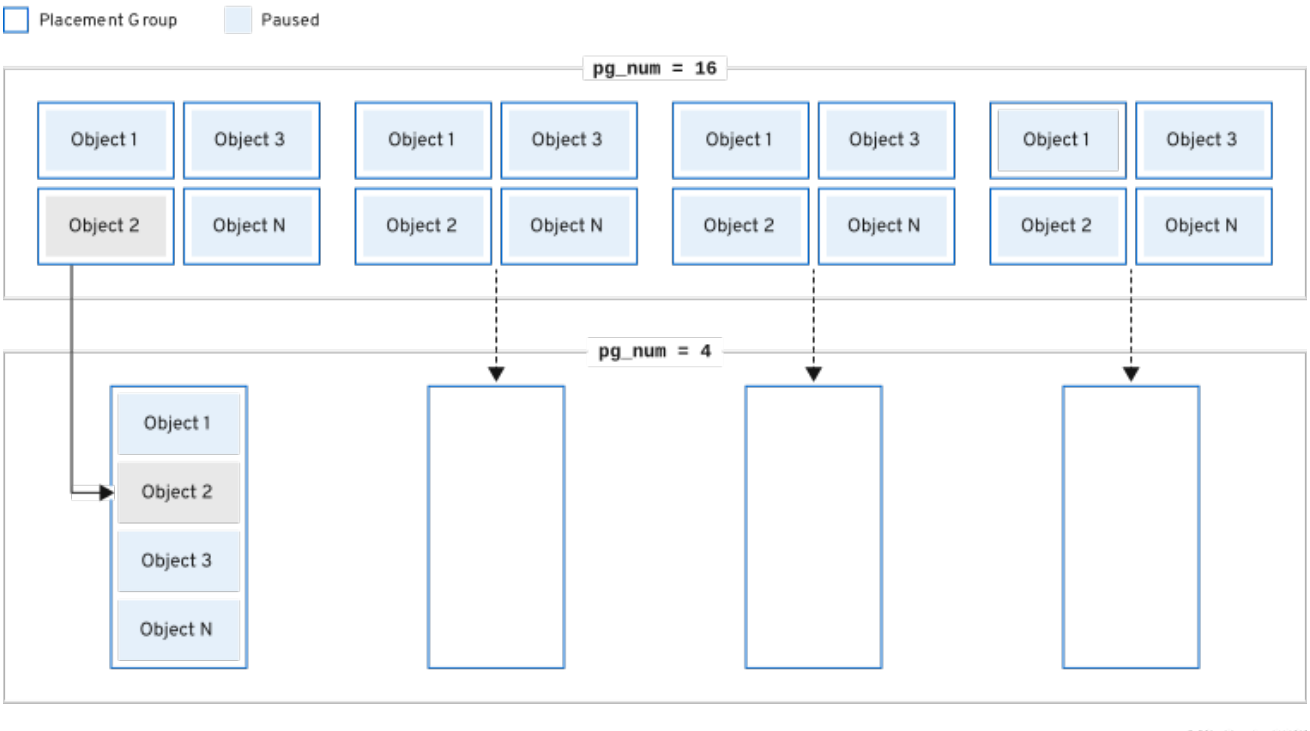
分割

Red Hat Ceph Storage は既存の配置グループ (PG) を小規模な PG に分割することができます。これにより、特定のプールの PG の合計数が増加します。既存の配置グループ (PG) を分割すると、ストレージ要件の増加に伴って、少数の Red Hat Ceph Storage クラスターを徐々にスケーリングできます。PG の自動スケーリング機能により、**pg_num** 値を増やすことができます。これにより、既存の PG がストレージクラスターを拡張するように分割されます。PG の自動スケーリング機能が無効な場合には、PG 分割プロセスを開始する **pg_num** 値を手動で増やすことができます。たとえば、**pg_num** の値を **4** から **16** に増やすと、4 つの部分に分割されます。**pg_num** 値を増やすと、**pgp_num** 値も増えますが、**pgp_num** 値は徐々に増加します。この段階の拡張は、オブジェクトデータを移行するため、ストレージクラスターのパフォーマンスおよびクライアントワークロードへの影響を最小限に抑えるために行われます。オブジェクトデータがシステムに大きな負荷が発生するためです。デフォルトでは、Ceph は misplaced の状態にあるオブジェクトデータの 5% を超えてキューを発行したり、移行したりしません。このデフォルトのパーセンテージは、**target_max_misplaced_ratio** オプションで調整できます。



マージ

Red Hat Ceph Storage は、既存の PG をサイズの大きい PG にマージすることもできるので、PG の合計が減少します。2 つの PG を 1 つにまとめると、プールの相対的な量が徐々に減少したときや、選択した PG の初期数が大きすぎたときに特に役に立ちます。PG のマージは便利ですが、複雑で繊細なプロセスでもあります。マージ時に、I/O の PG への一時停止が発生し、ストレージクラスターのパフォーマンスへの影響を最小限に抑えるために一度に PG を 1 つだけマージします。新しい **pg_num** 値に達するまで、Ceph はオブジェクトデータのマージにゆっくり機能します。



3.5.3. 配置グループの自動スケーリングモードの設定

Red Hat Ceph Storage クラスターの各プールには PG の **pg_autoscale_mode** プロパティーがあり、これを **off**、**on**、または **warn** に設定することができます。

- **off**: プールの自動スケーリングを無効にします。各プールに適切な PG 数を選択するのは管理者次第です。詳細は、配置グループ数 セクションを参照してください。
- **on**: 指定プールの PG 数の自動調整を有効にします。
- **warn**: PG 数の調整が必要な場合にヘルスアラートを示します。

注記

Red Hat Ceph Storage 5 以降のリリースでは、**pg_autoscale_mode** がデフォルトで **オン** になっています。アップグレードされたストレージクラスターは、既存の **pg_autoscale_mode** 設定を保持します。新しく作成されたプールでは、**pg_auto_scale** モードが **オン** になっています。PG カウントは自動的に調整され、**Ceph ステータス** が PG カウント調整中に回復中と表示される場合があります。

オートスケーラーは **bulk** フラグを使用して、PG の完全な補完で開始するプールを決定し、プール全体の使用率が均一でない場合にのみスケールダウンします。ただし、プールに **bulk** フラグがない場合、プールは最小限の PG で開始され、プールでの使用量が多い場合にのみ開始されます。

注記

オートスケーラーは重複するルートを特定し、そのルートが含まれるプールがスケーリングされないようにします。これは、root が重複しているとスケーリングプロセスに問題がある可能性があるためです。

手順

- 既存のプールで自動スケーリングを有効にします。

  構文

  ```
  ceph osd pool set POOL_NAME pg_autoscale_mode on
  ```

  例

  ```
  [ceph: root@host01 /]# ceph osd pool set testpool pg_autoscale_mode on
  ```

- 新規作成されたプールで自動スケーリングを有効にします。

  構文

  ```
  ceph config set global osd_pool_default_pg_autoscale_mode MODE
  ```

  例

  ```
  [ceph: root@host01 /]# ceph config set global osd_pool_default_pg_autoscale_mode on
  ```

- **bulk** フラグでプールを作成します。

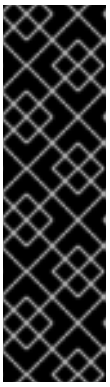
構文

```
ceph osd pool create POOL_NAME --bulk
```

例

```
[ceph: root@host01 /]#  ceph osd pool create testpool --bulk
```

- 既存プールの **bulk** フラグを設定または設定解除します。



重要

値は **true**、**false**、**1**、または **0** として書き込む必要があります。**1** は **true** に相当し、**0** は **false** に相当します。大文字と小文字が異なるか、他の内容で書かれている場合、エラーが発生します。

以下は、間違った構文で記述されたコマンドの例です。

```
[ceph: root@host01 /]# ceph osd pool set ec_pool_overwrite bulk True
Error EINVAL: expecting value 'true', 'false', '0', or '1'
```

構文

```
ceph osd pool set POOL_NAME bulk true/false/1/0
```

例

```
[ceph: root@host01 /]#  ceph osd pool set testpool bulk true
```

- 既存プールの **bulk** フラグを取得します。

構文

```
ceph osd pool get POOL_NAME bulk
```

例

```
[ceph: root@host01 /]# ceph osd pool get testpool bulk
bulk: true
```

3.5.4. 配置グループのスケーリングの推奨事項

プール、その相対的な使用率、およびストレージクラスター内の PG カウントに対する推奨の変更を表示できます。

前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。
- すべてのノードへの root レベルのアクセス。

手順

- 以下を使用して、各プール、その相対使用率、および推奨される変更をすべて表示できます。

```
[ceph: root@host01 /]# ceph osd pool autoscale-status
```

出力は以下のようになります。

```
POOL                    SIZE  TARGET SIZE  RATE  RAW CAPACITY   RATIO  TARGET RATIO
EFFECTIVE RATIO  BIAS  PG_NUM  NEW PG_NUM  AUTOSCALE  BULK
device_health_metrics      0               3.0        374.9G  0.0000                                  1.0       1
on         False
cephfs.cephfs.meta     24632               3.0        374.9G  0.0000                                  4.0
32              on         False
cephfs.cephfs.data         0               3.0        374.9G  0.0000                                  1.0      32
on         False
.rgw.root               1323               3.0        374.9G  0.0000                                  1.0      32
on         False
default.rgw.log         3702               3.0        374.9G  0.0000                                  1.0      32
on         False
default.rgw.control        0               3.0        374.9G  0.0000                                  1.0      32
on         False
default.rgw.meta         382               3.0        374.9G  0.0000                                  4.0       8
on         False
```

SIZE は、プールに保存されているデータ量です。

TARGET SIZE は、管理者が指定したデータ量を指しますが、最終的にこのプールに格納されることが予想されます。システムは、計算には、 2 つの値の大きいほうを使用します。

RATE は、プールが使用する RAW ストレージ容量を決定するプールの乗数です。たとえば、**3** つのレプリカプールは **3.0** の比率を持ち、**k=4,m=2** イレイジャーコードプールの比率は **1.5** になります。

RAW CAPACITY は、プールのデータを保存する OSD 上の RAW ストレージ容量の合計量です。

RATIO は、プールが消費している合計容量の比率です。つまり、ratio = size * rate / raw capacity になります。

TARGET RATIO (存在する場合) は、ターゲット比率が設定された他のプールと相対的にプールが消費することが予想されるストレージの比率です。ターゲットサイズバイトと比率の両方が指定される場合、比率が優先されます。プールの作成時に指定されていない限り、**TARGETRATIO** のデフォルト値は **0** です。プールで指定する **--target_ratio** が多いほど、プールで必要な PG は大きくなります。

EFFECTIVE RATIO は、次の 2 つの方法で調整した後の目標比率です。1 (目標サイズが設定されたプールで使用されると予想される容量を差し引く)。2. ターゲット比率が設定されたプール間でターゲット比率を正規化し、残りの領域を結合します。たとえば、**target ratio** が 1.0 の 4 つのプールの **effective ratio** は 0.25 になります。システムは、実際の比率が大きい場合と、その計算に効果的な比率を使用します。

BIAS は、特定のプールで必要とする PG 数に関する事前情報に基づいて、プールの PG を手動で調整するための乗数として使用されます。デフォルトでは、プールの作成時に指定されていない限り、値は 1.0 です。プールに指定する **--bias** が多いほど、プールで必要な PG は大きくなります。

PG_NUM は、プールの現在の PG 数、または **pg_num** の変更が進行中である場合にプールが現在操作している PG 数です。**NEW PG_NUM** が存在する場合は、推奨される PG 数 (**pg_num**) です。これは常に 2 の累乗となり、推奨される値は現在の値のみによって 3 倍以上になります。

AUTOSCALE はプール **pg_autoscale_mode** で、**on**、**off**、または **warn** のいずれかになります。

BULK は、どのプールが PG の完全な補完で開始するかを決定するために使用されます。**BULK** は、プール全体の使用率が均等でない場合にのみスケールダウンします。プールにこのフラグがない場合、プールは最小量の PG で開始され、プールの使用量が増加した場合にのみ使用されます。

BULK 値は **true**、**false**、**1**、または **0** です。**1** は **true** に相当し、**0** は **false** に相当します。デフォルト値は **false** です。

BULK 値は、プールの作成中または作成後に設定します。

バルクフラグの使用の詳細は、プールの作成 および 配置グループの自動スケールモードの設定 を参照してください。

3.5.5. 配置グループの自動スケーリングの設定

クラスターの使用状況に基づいて PG を自動的にスケーリングできるようにするのは、PG をスケーリングする最も簡単な方法です。Red Hat Ceph Storage は、利用可能なストレージと、全システムにおける PG のターゲット数を取得して、各プールに保存されたデータ量を比較し、それに応じて PG を評価します。このコマンドは、現在の PG 数 (**pg_num**) が計算または提案された PG 数から 3 倍以上ずれているプールにのみ変更を加えます。

各 OSD の PG のターゲット数は、**mon_target_pg_per_osd** 設定に基づいています。デフォルト値は **100** に設定されています。

手順

- **mon_target_pg_per_osd** を調整するには、以下を実行します。

  構文

  ```
  ceph config set global mon_target_pg_per_osd number
  ```

  以下に例を示します。

  ```
  [ceph: root@host01 /]# ceph config set global mon_target_pg_per_osd 150
  ```

3.5.6. noautoscale フラグの更新

すべてのプールに対して同時にオートスケーラーを有効または無効にする場合は、**noautoscale** グローバルフラグを使用できます。このグローバルフラグは、一部の OSD がバウンスされたとき、またはクラスターがメンテナンス中、ストレージクラスターのアップグレード中に役立ちます。アクティビティーの前にフラグを設定し、アクティビティーが完了したらフラグを解除できます。

デフォルトでは、**noautoscale** フラグは **off** に設定されています。このフラグが設定されている場合には、すべてのプールで **pg_autoscale_mode** が **オフ** になり、すべてのプールでオートスケーラーが無効になります。

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。

手順

1. **noautoscale** フラグの値を取得します。

   例

   ```
   [ceph: root@host01 /]# ceph osd pool get noautoscale
   ```

2. アクティビティーの前に **noautoscale** フラグを設定します。

   例

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

3. アクティビティーの完了時に **noautoscale** フラグの設定を解除します。

   例

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

3.6. ターゲットプールサイズの指定

新規作成されたプールは、クラスター容量の合計を少少なく、システムが PG の数を必要とするシステムに表示されます。ただし、ほとんどの場合、クラスター管理者は、どのプールが時間とともにシステム容量を消費することを認識します。Red Hat Ceph Storage への **ターゲットサイズ** として知られるこの情報を提供する場合、このようなプールは最初からより適切な数の PG (**pg_num**) を使用できます。このアプローチは、調整を行う際に、**pg_num** における後続の変更やデータの移動に関連するオーバーヘッドを防ぎます。

プールの **target size** は、以下の方法で指定できます。

- 「プールの絶対サイズ (バイト単位) を使用して target size を設定します。」
- 「クラスター合計容量を使用したターゲットサイズの指定」

3.6.1. プールの絶対サイズ (バイト単位) を使用して target size を設定します。

手順

1. プールの絶対サイズ (バイト単位) を使用して **target size** を設定します。

   ```
   ceph osd pool set pool-name target_size_bytes value
   ```

   たとえば、**mypool** が 100T の領域を消費することが予想されるようにシステムに指示します。

   ```
   $ ceph osd pool set mypool target_size_bytes 100T
   ```

また、任意の **--target-size-bytes <bytes>** 引数を **ceph osd pool create** コマンドに追加すると、作成時にプールのターゲットサイズを設定することもできます。

3.6.2. クラスター合計容量を使用したターゲットサイズの指定

手順

1. クラスター容量の合計の比率を使用して **target size** を設定します。

   構文

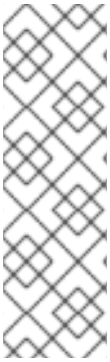
   ```
   ceph osd pool set pool-name target_size_ratio ratio
   ```

   以下に例を示します。

   ```
   [ceph: root@host01 /]# ceph osd pool set mypool target_size_ratio 1.0
   ```

   システムに、**target_size_ratio** が設定された他のプールと比較して、プール **mypool** が 1.0 を消費することが予想されることをシステムに指示します。**mypool** がクラスター内の唯一のプールである場合、これは、合計容量の 100% が予想される使用を意味します。**target_size_ratio** が 1.0 である 2 番目のプールがある場合、両方のプールはクラスター容量の 50% の使用を想定します。

また、任意の **--target-size-ratio <ratio>** 引数を **ceph osd pool create** コマンドに追加すると、作成時にプールのターゲットサイズを設定することもできます。



注記

不可能なターゲットサイズ値 (クラスターの合計よりも大きな容量、または 1.0 を超える合計の割合) を指定した場合、クラスターは **POOL_TARGET_SIZE_RATIO_OVERCOMMITTED** または **POOL_TARGET_SIZE_BYTES_OVERCOMMITTED** の警告を発生させます。

プールに **target_size_ratio** と **target_size_bytes** の両方を指定すると、クラスターは比率のみを考慮し、**POOL_HAS_TARGET_SIZE_BYTES_AND_RATIO** 正常性の警告を出します。

3.7. 配置グループのコマンドラインインターフェイス

ceph CLI では、プールの配置グループ数の設定および取得、PG マップの表示、PG の統計の取得を行うことができます。

3.7.1. プール内の配置グループ数の設定

プール内の配置グループの数を設定するには、プールの作成時に配置グループの数を指定する必要があります。詳細は、プールの作成 を参照してください。プールに配置グループを設定したら、配置グループの数を増やすことができます (ただし、配置グループの数を減らすことはできません)。配置グループの数を増やすには、以下のコマンドを実行します。

構文

```
ceph osd pool set POOL_NAME pg_num PG_NUM
```

配置グループの数を増やしたら、クラスターがリバランスする前に、配置 (**pgp_num**) の配置グループの数も増やす必要があります。**pgp_num** は **pg_num** と同じである必要があります。配置の配置グループの数を増やすには、以下のコマンドを実行します。

構文

-

```
ceph osd pool set POOL_NAME pgp_num PGP_NUM
```

3.7.2. プール内の配置グループ数の取得

プール内の配置グループの数を取得するには、以下のコマンドを実行します。

構文

```
ceph osd pool get POOL_NAME pg_num
```

3.7.3. 配置グループの統計の取得

ストレージクラスター内の配置グループの統計を取得するには、以下を実行します。

構文

```
ceph pg dump [--format FORMAT]
```

有効な形式は **plain** (デフォルト) および **json** です。

3.7.4. スタックした配置グループの統計の取得

指定された状態で固まったすべての配置グループを取得するには、以下を実行します。

構文

```
ceph pg dump_stuck {inactive|unclean|stale|undersized|degraded
[inactive|unclean|stale|undersized|degraded...]} INTERVAL
```

Inactive 配置グループは、最新のデータを持つ OSD が up で in になることを待っているため、読み取りや書き込みを処理できません。

Unclean 配置グループには、希望する回数を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。

Stale 配置グループは不明な状態にあります それをホストする OSD は、しばらくモニタークラスターに対して報告されていない OSD です (**mon_osd_report_timeout** で設定されます)。

有効な形式は **plain** (デフォルト) および **json** です。このしきい値は、返される統計に含める前に、配置グループがス詰まった最小秒数を定義します (デフォルトは 300 秒)。

3.7.5. 配置グループマップの取得

特定の配置グループの配置グループマップを取得するには、以下のコマンドを実行します。

構文

```
ceph pg map PG_ID
```

例

```
[ceph: root@host01 /]# ceph pg map 1.6c
```

-

Ceph は配置グループマップ、配置グループ、および OSD ステータスを返します。

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

3.7.6. 配置グループのスクラブ

配置グループをスクラブするには、以下のコマンドを実行します。

構文

```
ceph pg scrub PG_ID
```

Ceph はプライマリーノードとレプリカノードを確認し、配置グループ内の全オブジェクトのカタログを生成し、オブジェクトが見つからないか、一致しないか、その内容の一貫性を保つようにします。レプリカがすべて一致したことを想定すると、最終的なセマンティックスイープにより、すべてのスナップショット関連のオブジェクトメタデータの一貫性が確保されます。エラーはログにより報告されます。

3.7.7. unfound オブジェクトのマーク

クラスターが 1 つ以上のオブジェクトが失われ、失われたデータの検索を破棄した場合、失われたオブジェクトを **lost** とマークする必要があります。

可能なロケーションがすべてクエリーされ、オブジェクトが依然として失われている場合は、失われたオブジェクトは諦めるしかありません。この障害には、書き込み自体が復旧する前に実行された書き込みについて、クラスターが認識できるようにする、障害上の組み合わせが発生したことが考えられます。

現時点で、サポートされる唯一のオプションは revert です。これはオブジェクトの以前のバージョンにロールバックするか、(新しいオブジェクトの場合は) 完全にロールバックします。unfound オブジェクトを lost とマークするには、以下を実行します。

構文

```
ceph pg PG_ID mark_unfound_lost revert|delete
```

重要

オブジェクトが存在すると想定されるアプリケーションが同じ場合があるため、この機能は注意して使用してください。

第4章 プールの概要

Ceph クライアントは、データをプールに保存します。プールの作成時に、クライアントがデータを保存するための I/O インターフェイスを作成します。

Ceph クライアント、つまりブロックデバイス、ゲートウェイ、その他の観点から見ると、Ceph Storage クラスターとの対話は非常に簡単です。

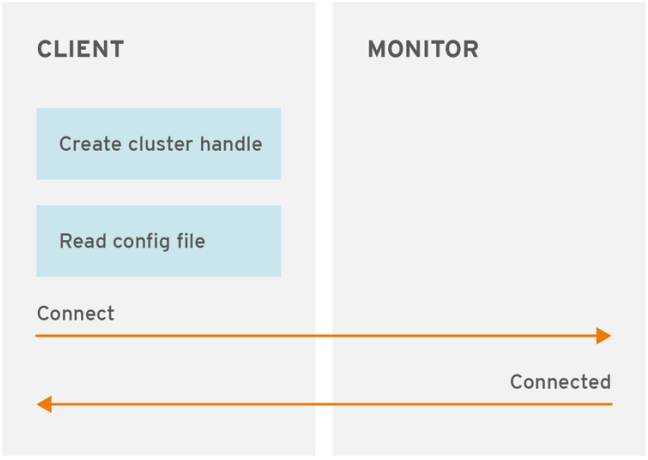
- クラスターハンドルを作成します。
- クラスターハンドルをクラスターに接続します。
- オブジェクトとその拡張属性を読み書きするための I/O コンテキストを作成します。

クラスターハンドルの作成とクラスターへの接続

Ceph Storage クラスターに接続するには、Ceph クライアントに次の詳細が必要です。

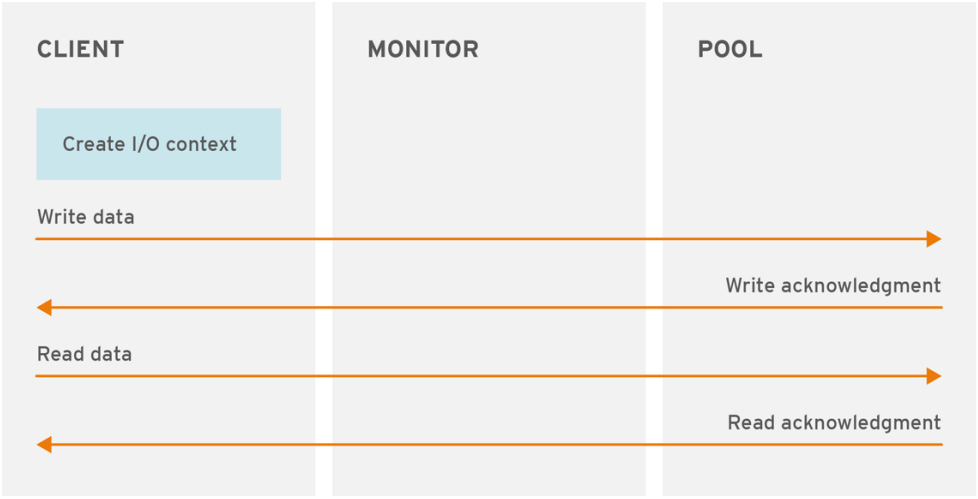
- クラスター名 (デフォルトでは Ceph) - 曖昧に聞こえるため、通常は使用しません。
- 初期モニターアドレス。

通常、Ceph クライアントは Ceph 設定ファイルのデフォルトパスを使用してパラメーターを取得し、ファイルからファイルを読み取りますが、コマンドラインでパラメーターを指定することもできます。Ceph クライアントは、ユーザー名と秘密鍵も提供します。認証はデフォルトで **On** になっています。次に、クライアントは Ceph monitor クラスターに接続し、モニター、OSD、およびプールを含むクラスターマップの最新コピーを取得します。



プール I/O コンテキストの作成

データを読み書きするために、Ceph クライアントは Ceph Storage クラスター内の特定のプールへの I/O コンテキストを作成します。指定したユーザーにプールのパーミッションがある場合は、Ceph クライアントは指定されたプールから読み取り/書き込みを行うことができます。



Ceph のアーキテクチャーを使用することで、ストレージクラスターは、プール名を指定して簡単に定義し、I/O コンテキストの作成で簡単に定義するストレージストラテジーのいずれかをクライアントが選択できるように、ストレージクラスターを Ceph クライアントに提供することができます。ストレージストラテジーはすべて、容量およびパフォーマンスにおいて Ceph クライアントを認識しません。同様に、Ceph クライアントの複雑性 (例: ブロックデバイス表現へのオブジェクトのマッピング、S3/Swift RESTful サービスの提供) は Ceph ストレージクラスターに見えません。

プールは、復元力、配置グループ、CRUSH ルール、およびクォータを提供します。

- **耐障害性:** データを損失せずに失敗した OSD の数を設定できます。複製されたプールの場合、これはオブジェクトのコピーまたはレプリカの任意数です。通常の設定では、オブジェクトと 1 つの追加コピー (例: **size = 2**) が保存されますが、コピーまたはレプリカの数を決定できます。イレイジャーコードプールの場合は、コーディングしたチャンクの数です (例: **イレイジャーコードプロファイル** の **m=2**)。

- **配置グループ**: プールの配置グループの数を設定できます。典型的な設定では、OSD ごとに約 50-100 の配置グループを使用して、最高のコンピューティングリソースを使用せずに最適なバランスを提供します。複数のプールを設定する場合は、全体としてプールとクラスターの両方に妥当な配置グループ数を設定するように注意してください。

- **CRUSH ルール**: プールにデータをプールに保存すると、CRUSH ルールがプールにマッピングされた CRUSH ルールにより、CRUSH が各オブジェクトとそのレプリカ (またはイレイジャーコード化されたプールのチャンク) の配置のルールを特定できます。プールにカスタム CRUSH ルールを作成できます。

- **クォータ**: **ceph osd pool set-quota** コマンドを使用してプールにクォータを設定すると、指定したプールに保存されるオブジェクトの最大数または最大バイト数が制限される場合があります。

4.1. プールおよびストレージストラテジーの概要

プールを管理するために、プールのリストを表示、作成、および削除できます。各プールの使用状況の統計を表示することもできます。

4.2. プールの一覧表示

クラスターのプールを一覧表示します。

例

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

4.3. プールの作成

プールを作成する前に、**設定ガイド** を参照してください。

デフォルト値はニーズに合う必要がないため、配置グループの数のデフォルト値を調整することを推奨します。

例

```
[ceph: root@host01 /]# ceph config set global osd_pool_default_pg_num 250
[ceph: root@host01 /]# ceph config set global osd_pool_default_pgp_num 250
```

複製されたプールを作成します。

構文

```
ceph osd pool create POOL_NAME PG_NUM PGP_NUM [replicated] \
     [CRUSH_RULE_NAME] [EXPECTED_NUMBER_OBJECTS]
```

イレイジャーコーディングされたプールを作成します。

構文

```
ceph osd pool create POOL_NAME PG_NUM PGP_NUM erasure \
     [ERASURE_CODE_PROFILE] [CRUSH_RULE_NAME] [EXPECTED_NUMBER_OBJECTS]
```

バルクプールを作成します。

構文

```
ceph osd pool create POOL_NAME [--bulk]
```

ここでは、以下のようになります。

POOL_NAME

説明

プールの名前。一意でなければなりません。

型

String

必須

必須です。指定しない場合は、デフォルト値に設定されます。

デフォルト

ceph

PG_NUM

説明

プールの現在の配置グループ数。適切な数の計算の詳細は、**配置グループ** セクションおよび Ceph Placement Groups (PGs) per Pool Calculatorを参照してください。デフォルト値 **8** は、ほとんどのシステムには適していません。

型

整数

必須

Yes

デフォルト

8

PGP_NUM

説明

配置目的の配置グループの合計数。この値は、配置グループ分割シナリオを除き、配置グループの合計数と同じでなければなりません。

型

整数

必須

必須です。指定しない場合は、デフォルト値に設定されます。

デフォルト

8

replicated または erasure

説明

オブジェクトまたは **erasure** を複数維持することで失われた OSD から復旧するために **複製** することのできるプールタイプは、一般的な RAID5 機能を取得します。複製されたプールにはより多くの raw ストレージが必要であり、すべての Ceph 操作が実装されます。消去コード化されたプールにより必要な raw ストレージは少なくなりますが、この場合利用可能な操作のサブセットのみが実装されます。

型

String

必須

いいえ

デフォルト

replicated

CRUSH_RULE_NAME

説明

プールの CRUSH ルールの名前。このルールが存在する **必要があります**。レプリケートされたプールの場合、名前は **osd_pool_default_crush_rule** 設定で指定されたルールになります。イレイジャーコーディングされたプールの場合は、デフォルトのイレイジャーコードプロファイルまたは **POOL_NAME** を指定すると、名前が **erasure-code** になります。ルールがまだ存在しない場合、Ceph は指定された名前でこのルールを暗黙的に作成します。

型

String

必須

いいえ

デフォルト

イレイジャーコーディングされたプールに **erasure-code** を使用します。複製されたプールの場合は、Ceph 設定からの **osd_pool_default_crush_rule** 変数の値を使用します。

EXPECTED_NUMBER_OBJECTS

説明

プールに必要なオブジェクト数Ceph は、実行時ディレクトリー分割の実行による遅延の影響を回避するために、プールの作成時に配置グループを分割します。

型

整数

必須

いいえ

デフォルト

0、プール作成時に分割なし。

ERASURE_CODE_PROFILE

説明

イレイジャーコーディングされたプールのみの場合。イレイジャーコードプロファイルを使用します。これは、Ceph 設定ファイルの **osd erasure-code-profile set** 変数で定義されている既存のプロファイルでなければなりません。詳細は、イレイジャーコードプロファイル セクションを参照してください。

型

String

必須

いいえ

プールの作成時に、配置グループの数を妥当な値に設定します (例: **100**)。OSD ごとの配置グループの総数を考慮してください。配置グループは計算コストが高いため、多数の配置グループを持つプールが多数ある場合 (たとえば、それぞれ 100 個の配置グループを持つ 50 個のプール)、パフォーマンスが低下します。終了点は、OSD ホストのパワーによって異なります。

関連情報

プールの適切な配置グループ数を計算する方法は、配置グループ セクションおよび Ceph Placement Groups (PGs) per Pool Calculator を参照してください。

4.4. プールクォータの設定

プールあたりの最大バイト数と最大オブジェクト数のプールクォータを設定できます。

構文

```
ceph osd pool set-quota POOL_NAME [max_objects OBJECT_COUNT] [max_bytes BYTES]
```

例

```
[ceph: root@host01 /]# ceph osd pool set-quota data max_objects 10000
```

クォータを削除するには、その値を **0** に設定します。

注記

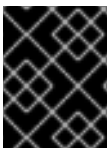
インフライト書き込み操作は、Ceph がプールの使用をクラスター全体の伝播するまで、短時間にプールクォータを過剰に実行する可能性があります。これは通常の動作です。インフライト書き込み操作でプールクォータを適用すると、パフォーマンスが大幅に低下します。

4.5. プールの削除

プールを削除します。

構文

```
ceph osd pool delete POOL_NAME [POOL_NAME --yes-i-really-really-mean-it]
```



重要

デフォルトでは、データ保護のため、ストレージ管理者はプールを削除できません。プールを削除する前に **mon_allow_pool_delete** 設定オプションを設定します。

プールに独自のルールがある場合は、プールの削除後に削除することを検討してください。プールにユーザー自体の使用を厳密に使用する場合は、プールの削除後にそれらのユーザーを削除することを検討してください。

4.6. プールの名前変更

プールの名前を変更します。

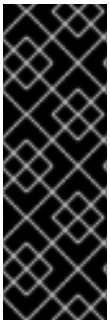
構文

```
ceph osd pool rename CURRENT_POOL_NAME NEW_POOL_NAME
```

プールの名前を変更し、認証されたユーザーにプールごとの機能がある場合は、ユーザーの機能 (上限) を新しいプール名で更新する必要があります。

4.7. プールの移行

すべてのオブジェクトをあるプールから別のプールに移行することが必要な場合があります。これは、特定のプールでは変更できないパラメーターを変更する必要がある場合などに行われます。たとえば、プールの配置グループの数を減らす必要がある場合です。



重要

ワークロードが Ceph ブロックデバイスイメージ **のみ** を使用している場合は、**Red Hat Ceph Storage ブロックデバイスガイド** に記載されているプールの移動および移行の手順に従ってください。

- プール間のイメージの移動
- プールの移行

Ceph ブロックデバイスについて説明している移行方法は、このドキュメントに記載の移行方法よりも推奨されます。cppool を使用すると、すべてのスナップショットとスナップショット関連のメタデータが保存されないため、データの不正確なコピーが作成されます。たとえば、RBD プールをコピーしても、イメージは完全にはコピーされません。この場合、スナップは存在しないため、正しく機能しません。cppool は、一部の librados ユーザーが依存する可能性のある **user_version** フィールドも保存しません。

プールの移行が必要で、ユーザーのワークロードに Ceph ブロックデバイス以外のイメージが含まれている場合は、ここに記載されている手順のいずれかを続行してください。

前提条件

- **rados cppool** コマンドを使用する場合は、以下のとおりです。
  - プールへの読み取り専用アクセスが必要です。
  - このコマンドは、librados によって消費される RBD イメージとそのスナップおよび **user_version** がない場合にのみ使用してください。
- ローカルドライブの RADOS コマンドを使用する場合は、十分なクラスター領域が利用可能であることを確認してください。プールのレプリケーション係数に従って、2 つ、3 つ、またはそれ以上のデータのコピーが存在します。

手順

方法 1 - 推奨される直接的な方法

rados cppool コマンドを使用してすべてのオブジェクトをコピーします。

重要

コピー中はプールへの読み取り専用アクセスが必要です。

構文

```
ceph osd pool create NEW_POOL PG_NUM [ <other new pool parameters> ]
rados cppool SOURCE_POOL NEW_POOL
ceph osd pool rename SOURCE_POOL NEW_SOURCE_POOL_NAME
ceph osd pool rename NEW_POOL SOURCE_POOL
```

例

```
[ceph: root@host01 /]# ceph osd pool create pool1 250
[ceph: root@host01 /]# rados cppool pool2 pool1
[ceph: root@host01 /]# ceph osd pool rename pool2 pool3
```

```
[ceph: root@host01 /]# ceph osd pool rename pool1 pool2
```

方法 2 - ローカルドライブを使用する方法

1. **rados export** および **rados import** コマンドと一時ローカルディレクトリーを使用して、エクスポートされたすべてのデータを保存します。

   構文

   ```
   ceph osd pool create NEW_POOL PG_NUM [ <other new pool parameters> ]
   rados export --create SOURCE_POOL FILE_PATH
   rados import FILE_PATH NEW_POOL
   ```

   例

   ```
   [ceph: root@host01 /]# ceph osd pool create pool1 250
   [ceph: root@host01 /]# rados export --create pool2 <path of export file>
   [ceph: root@host01 /]# rados import <path of export file> pool1
   ```

2. 必須。ソースプールへのすべての I/O を停止します。

3. 必須。変更されたすべてのオブジェクトを再同期します。

   構文

   ```
   rados export --workers 5 SOURCE_POOL FILE_PATH
   rados import --workers 5 FILE_PATH NEW_POOL
   ```

   例

   ```
   [ceph: root@host01 /]# rados export --workers 5 pool2 <path of export file>
   [ceph: root@host01 /]# rados import --workers 5 <path of export file> pool1
   ```

4.8. プールの統計の表示

プールの使用率統計を表示します。

例

```
[ceph: root@host01 /] rados df
```

4.9. プール値の設定

プールに値を設定します。

構文

```
ceph osd pool set POOL_NAME KEY VALUE
```

プール値 セクションに、設定可能なキーと値のペアをすべて示します。

4.10. プール値の取得

プールから値を取得します。

構文

```
ceph osd pool get POOL_NAME KEY
```

プールの値 セクションで取得できるすべてのキーと値のペアのリストを表示できます。

4.11. クライアントアプリケーションの有効化

Red Hat Ceph Storage は、未承認のクライアントタイプがプールにデータを書き込むのを防ぐために、プールに対する保護を強化します。つまり、システム管理者は、プールが Ceph Block Device、Ceph Object Gateway、Ceph Filesystem、またはカスタムアプリケーションから I/O 操作を受信するように指定する必要があります。

クライアントアプリケーションがプールで I/O 操作を実行できるようにします。

構文

```
ceph osd pool application enable POOL_NAME APP {--yes-i-really-mean-it}
```

APP には以下を指定します。

- Ceph Filesystem 用の **cephfs**。
- **Ceph** ブロックデバイスの rbd。
- **Ceph** Object Gateway の rgw。

注記

カスタムアプリケーションの場合は、別の **APP** 値を指定します。

重要

有効ではないプールは、**HEALTH_WARN** ステータスを生成します。そのようなシナリオでは、**ceph health detail -f json-pretty** の出力は次のようになります。

```
{
    "checks": {
        "POOL_APP_NOT_ENABLED": {
            "severity": "HEALTH_WARN",
            "summary": {
                "message": "application not enabled on 1 pool(s)"
            },
            "detail": [
                {
                    "message": "application not enabled on pool '_POOL_NAME_'"
                },
                {
                    "message": "use 'ceph osd pool application enable _POOL_NAME_
_APP_', where _APP_ is 'cephfs', 'rbd', 'rgw', or freeform for custom applications."
                }
            ]
        }
    },
    "status": "HEALTH_WARN",
    "overall_status": "HEALTH_WARN",
    "detail": [
        "'ceph health' JSON format has changed in luminous. If you see this your
monitoring system is scraping the wrong fields. Disable this with 'mon health
preluminous compat warning = false'"
    ]
}
```

注記

rbd pool init POOL_NAME を使用して、Ceph ブロックデバイスのプールを初期化します。

4.12. クライアントアプリケーションの無効化

クライアントアプリケーションがプールで I/O 操作を実行できないようにします。

構文

```
ceph osd pool application disable POOL_NAME APP {--yes-i-really-mean-it}
```

APP には以下を指定します。

- Ceph Filesystem 用の **cephfs**。
- **Ceph** ブロックデバイスの rbd。
- **Ceph** Object Gateway の rgw。

注記

カスタムアプリケーションの場合は、別の **APP** 値を指定します。

4.13. アプリケーションメタデータの設定

クライアントアプリケーションの属性を記述するキーと値のペアを設定する機能を提供します。

プールにクライアントアプリケーションのメタデータを設定します。

構文

```
ceph osd pool application set POOL_NAME APP KEY
```

APP には以下を指定します。

- Ceph Filesystem 用の **cephfs**。
- Ceph ブロックデバイス用の **rbd**
- Ceph Object Gateway 用の **rgw**

注記

カスタムアプリケーションの場合は、別の **APP** 値を指定します。

4.14. アプリケーションメタデータの削除

プールのクライアントアプリケーションメタデータを削除します。

構文

```
ceph osd pool application rm POOL_NAME APP KEY
```

APP には以下を指定します。

- Ceph Filesystem 用の **cephfs**。
- Ceph ブロックデバイス用の **rbd**
- Ceph Object Gateway 用の **rgw**

注記

カスタムアプリケーションの場合は、別の **APP** 値を指定します。

4.15. オブジェクトレプリカ数の設定

複製されたプールのオブジェクトレプリカの数を設定します。

構文

```
ceph osd pool set POOL_NAME size NUMBER_OF_REPLICAS
```

このコマンドは、プールごとに実行できます。

重要

NUMBER_OF_REPLICAS パラメーターには、オブジェクト自体が含まれます。オブジェクトの合計 3 つのインスタンスについて、オブジェクトとオブジェクトの 2 つのコピーを含める場合は、**3** を指定します。

例

```
[ceph: root@host01 /]# ceph osd pool set data size 3
```

注記

オブジェクトは、**pool size** 設定での指定よりも少ないレプリカを持つ低下モードで I/O 操作を受け入れることができます。I/O に必要なレプリカの最小数を設定するには、**min_size** 設定を使用します。

例

```
[ceph: root@host01 /]# ceph osd pool set data min_size 2
```

これにより、データプール内のオブジェクトは、**min_size** 設定で指定されたよりも少ないレプリカを持つ I/O を受信しないようになります。

4.16. オブジェクトレプリカ数の取得

オブジェクトレプリカの数を取得します。

例

```
[ceph: roo@host01 /]# ceph osd dump | grep 'replicated size'
```

Ceph はプールを一覧表示し、**replicated size** 属性を強調表示します。デフォルトでは、Ceph はオブジェクトのレプリカを 2 つ (つまり合計 3 つ、またはサイズ **3**) 作成します。

4.17. プール値

以下のリストには、設定または取得可能なキーと値のペアが含まれます。詳細は、プール値の設定 および プール値の取得 を参照してください。

size

説明

プール内のオブジェクトのレプリカ数を指定します。詳細は、オブジェクトレプリカ数の設定 セクションを参照してください。レプリケートされたプールにのみ適用可能です。

型

整数

min_size

説明

I/O に必要なレプリカの最小数を指定します。詳細は、オブジェクトレプリカ数の設定 セクションを参照してください。イレイジャーコーディングされたプールの場合、**k** より大きい値に設定する必要があります。I/O が値 **k** で許可されている場合は、冗長性がなく、永続的な OSD 障害が発生した場合にデータが失われます。詳細は、イレイジャーコードプールの概要 を参照してください。

型

整数

crash_replay_interval

説明

クライアントが確認応答を再生できますが、コミットされていないリクエストを再実行できる秒数を指定します。

型

整数

pg-num

説明

プールの現在の配置グループ数。適切な数を計算する方法は、Red Hat Ceph Storage の **設定ガイド** の **プール、配置グループ、および CRUSH 設定リファレンス** セクションを参照してください。デフォルト値 **8** は、ほとんどのシステムには適していません。

型

整数

必須

必須です。

デフォルト

8

pgp-num

説明

配置目的の配置グループの合計数。これは、配置グループ分割シナリオを除き、**配置グループの合計数と同じでなければなりません。**

型

整数

必須

必須です。指定されていない場合は、デフォルトの Ceph 設定値を選択します。

デフォルト

8

Valid Range

pg_num 変数で指定された値以下。

crush_rule

説明

クラスター内のオブジェクトの配置にマッピングするために使用するルール。

型

String

hashpspool

説明

指定プールの **HASHPSPOOL** フラグを有効または無効にします。このオプションを有効にすると、プールのハッシュおよび配置グループのマッピングが変更され、プールと配置グループの重複が改善されます。

型

整数

Valid Range

1 はフラグを有効にし、**0** はフラグを無効にします。

重要

多数の OSD およびデータを持つクラスターの実稼働プールでは、このオプションを有効にしないでください。プール内のすべての配置グループを再マッピングする必要があります。これにより、データの移動が大幅に低下します。

fast_read

説明

イレイジャーコーディングを使用するプールでは、このフラグが有効にされている場合、読み取り要求がすべてのシャードに対して読み取り要求を発行し、クライアントを提供するために十分なシャードを受信するまで待機します。**jerasure** プラグインおよび **isa erasure** プラグインの場合は、最初の K 応答が返されると、クライアントのリクエストは応答からデコードされたデータを即座に使用します。これにより、パフォーマンスを向上させるためにリソースを確保するのに役立ちます。現在、このフラグはイレイジャーコーディングプールにのみサポートされています。

型

Boolean

デフォルト

0

allow_ec_overwrites

説明

イレイジャーコードのプールへの書き込みがオブジェクトの一部を更新できるかどうかで、Ceph ファイルシステムおよび Ceph ブロックデバイスがこれを使用できるようにします。

型

Boolean

compression_algorithm

説明

BlueStore ストレージバックエンドで使用するインライン圧縮アルゴリズムを設定します。この設定は、**bluestore_compression_algorithm** 設定を上書きします。

型

String

有効の設定

lz4、**snappy**、**zlib**、**zstd**

compression_mode

説明

BlueStore ストレージバックエンドのインライン圧縮アルゴリズムのポリシーを設定します。この設定は、**bluestore_compression_mode** 設定を上書きします。

型

String

有効の設定

none、**passive**、**aggressive**、**force**

compression_min_blob_size

説明

BlueStore は、このサイズより小さいチャンクを圧縮しません。この設定は、**bluestore_compression_min_blob_size** 設定を上書きします。

型

未署名の整数

compression_max_blob_size

説明

BlueStore は、データを圧縮する前に、このサイズより大きいチャンクを **compression_max_blob_size** の小さなブロブに分割します。

型

未署名の整数

nodelete

説明

指定されたプールで **NODELETE** フラグを設定または解除します。

型

整数

Valid Range

1 はフラグを設定します。**0** はフラグの設定を解除します。

nopgchange

説明

指定したプールに **NOPGCHANGE** フラグを設定または設定解除します。

型

整数

Valid Range

1 はフラグを設定します。**0** はフラグの設定を解除します。

nosizechange

説明

指定したプールに **NOSIZECHANGE** フラグを設定または設定解除します。

型

整数

Valid Range

1 はフラグを設定します。**0** はフラグの設定を解除します。

write_fadvise_dontneed

説明

特定のプールの **WRITE_FADVISE_DONTNEED** フラグを設定または解除します。

型

整数

Valid Range

1 はフラグを設定します。**0** はフラグの設定を解除します。

noscrub

説明

指定したプールに **NOSCRUB** フラグを設定または設定解除します。

型

整数

Valid Range

1 はフラグを設定します。**0** はフラグの設定を解除します。

nodeep-scrub

説明

指定したプールに **NODEEP_SCRUB** フラグを設定または設定解除します。

型

整数

Valid Range

1 はフラグを設定します。**0** はフラグの設定を解除します。

scrub_min_interval

説明

負荷が低い際のプールスクラビングの最小間隔 (秒単位)。**0** の場合、Ceph は **osd_scrub_min_interval** の設定を使用します。

型

Double

デフォルト

0

scrub_max_interval

説明

クラスター負荷のプールスクラビングが最大の間隔 (秒単位)。**0** の場合、Ceph は **osd_scrub_max_interval** の設定を使用します。

型

Double

デフォルト

0

deep_scrub_interval

説明

プール 'deep' スクラビングの間隔 (秒単位)。**0** の場合、Ceph は **osd_deep_scrub_interval** の設定を使用します。

型

Double

デフォルト

0

第5章 イレイジャーコードプールの概要

Ceph はデフォルトで複製されたプールを使用します。これにより、Ceph はすべてのオブジェクトをプライマリー OSD ノードから 1 つ以上のセカンダリー OSD にコピーします。イレイジャーコーディングされたプールは、データの持続性を確保するのに必要なディスク容量を減らしますが、レプリケーションよりもコストが高くなります。

Ceph ストレージストラテジーには、データの持続性要件を定義します。データの持続性とは、データが失われることなく、1 つまたは複数の OSD の損失を持続させることができることを意味します。

Ceph は、データをプールに保存します。プールには 2 種類のプールがあります。

- **replicated**
- **erasure-coded**

イレイジャーコーディングは、Ceph ストレージクラスターにオブジェクトを大幅に格納する方法であり、イレイジャーコードアルゴリズムによりオブジェクトがデータチャンク (**k**)、およびコーディングチャンク (**m**) に分割され、これらのチャンクを異なる OSD に保存します。

OSD に障害が発生すると、Ceph は他の OSD から残りのデータ (**k**) およびコーディング (**m**) チャンクを取得し、イレイジャーコードアルゴリズムはこれらのチャンクからオブジェクトを復元します。

注記

Red Hat は、書き込みやデータの損失を防ぐために、イレイジャーコーディングされたプールの **min_size** を **K+1** 以上にすることを推奨します。

イレイジャーコーディングは、レプリケーションよりもストレージ容量をより効率的に使用します。n 個のレプリケーションアプローチは、オブジェクトの **n** 個のコピーを維持するのに対し (Ceph のデフォルトは 3x)、イレイジャーコーディングは **k** + **m** チャンクのみを保持します。たとえば、3 データと 2 つのブロックのチャンクは、元のオブジェクトの 1.5x のストレージ領域を使用します。

イレイジャーコーディングはレプリケーションと比べてストレージのオーバーヘッドが少なく、イレイジャーコードアルゴリズムは、オブジェクトへのアクセスや復旧時に、レプリケーションよりも多くの RAM および CPU を使用します。イレイジャーコーディングは、データストレージが永続的であり、耐障害性になければならないものの、高速な読み取り (たとえば、コールドマイグレーションストレージ、履歴レコードなど) を必要としない場合に役立ちます。

Ceph でイレイジャーコードがどのように機能するかの詳細は、Red Hat Ceph Storage 7 の **アーキテクチャーガイド** の Ceph イレイジャーコーディング セクションを参照してください。

k=2 および **m=2** を使用してクラスターを初期化する際に、Ceph は **デフォルト** のイレイジャーコードプロファイルを作成します。つまり、Ceph は 3 つの OSD (**k+m == 4**) にオブジェクトデータを分散し、Ceph がこれらの OSD のいずれかを、データを失うことなく失う可能性があることを意味します。イレイジャーコードのプロファイリングの詳細は、イレイジャーコードプロファイル セクションを参照してください。

重要

.rgw.buckets プールのみをイレイジャーコーディング済みとして設定し、その他のすべての Ceph Object Gateway プールをレプリケート済みとして設定すると、新しいバケットを作成しようとすると以下のエラーで失敗します。

```
set_req_state_err err_no=95 resorting to 500
```

このため、イレイジャーコーディングされたプールは **omap** 操作をサポートしません。特定の Ceph Object Gateway メタデータプールには **omap** サポートが必要です。

5.1. イレイジャーコーディングされたプールのサンプルの作成

イレイジャーコーディングプールを作成し、配置グループを指定します。

最も簡単なイレイジャーコードプールは RAID5 と同等で、少なくとも 4 台のホストが必要です。2+2 プロファイルを使用してイレイジャーコーディングされたプールを作成できます。

手順

1. 2+2 設定の 4 つのノード上のイレイジャーコーディングされたプールに対して次の設定を設定します。

   構文

   ```
   ceph config set mon mon_osd_down_out_subtree_limit host
   ceph config set osd osd_async_recovery_min_cost 1099511627776
   ```

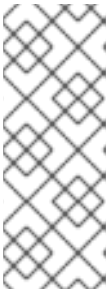
   重要

   一般に、イレイジャーコーディングプールではこれは必要ありません。

   重要

   非同期リカバリーのコストは、レプリカ上で遅れている PG ログエントリーの数と、失われたオブジェクトの数です。**osd_target_pg_log_entries_per_osd** は **30000** です。したがって、単一の PG を持つ OSD には **30000** のエントリーが存在する可能性があります。**osd_async_recovery_min_cost** は 64 ビットの整数であるため、2+2 設定の EC プールの場合は、**osd_async_recovery_min_cost** の値を **1099511627776** に設定します。

   注記

   4 つのノードを持つ EC クラスターの場合、K+M の値は 2+2 です。ノードに完全な障害が発生した場合、ノードは 4 つのチャンクとして回復されず、使用できるノードは 3 つだけになります。**mon_osd_down_out_subtree_limit** の値を **host** に設定すると、ホストのダウンシナリオ中に OSD がマークアウトされなくなり、データの再バランシングやノードが再び起動するまでの待機が防止されます。

2. 2+2 設定のイレイジャーコーディングされたプールの場合は、プロファイルを設定します。

   構文

```
ceph osd erasure-code-profile set ec22 k=2 m=2 crush-failure-domain=host
```

例

```
[ceph: root@host01 /]# ceph osd erasure-code-profile set ec22 k=2 m=2 crush-failure-domain=host

Pool : ceph osd pool create test-ec-22 erasure ec22
```

3. イレイジャーコーディングされたプールを作成します。

例

```
[ceph: root@host01 /]# ceph osd pool create ecpool 32 32 erasure

pool 'ecpool' created
$ echo ABCDEFGHI | rados --pool ecpool put NYAN -
$ rados --pool ecpool get NYAN -
ABCDEFGHI
```

32 は配置グループの数です。

5.2. イレイジャーコードプロファイル

Ceph は、**プロファイル** でイレイジャーコーディングされたプールを定義します。Ceph は、イレイジャーコーディングされたプールおよび関連する CRUSH ルールを作成する際にプロファイルを使用します。

Ceph は、クラスターを初期化する時にデフォルトのイレイジャーコードプロファイルを作成し、レプリケートされたプール内の 2 つのコピーと同じレベルの冗長性を提供します。このデフォルトのプロファイルは **k=2** と **m=2** を定義します。つまり、Ceph はオブジェクトデータを 4 つの OSD (**k+m=4**) に分散するため、これらの OSD の 1 つを失ってもデータは失われません。EC2+2 には、最小 4 ノード (推奨は 5 ノード) のデプロイメントフットプリントが必要で、1 つの OSD ノードの一時的な損失に対処できます。

デフォルトのプロファイルを表示するには、以下のコマンドを使用します。

```
$ ceph osd erasure-code-profile get default
k=2
m=2
plugin=jerasure
technique=reed_sol_van
```

Raw ストレージ要件を増加させずに冗長性を強化するために、新規プロファイルを作成できます。たとえば、**k=8** と **m=4** のプロファイルでは、12 個の (**k+m=12**) OSD オブジェクトを配布することで、4 個の (**m=4**) OSD が失われないようにすることができます。Ceph はオブジェクトを **8** つのチャンクに分割し、リカバリー用に **4** つのコーディングチャンクを計算します。たとえば、オブジェクトサイズが 8 MB の場合、各データチャンクが 1 MB で、各データチャンクのサイズは 1 MB であり、各データチャンクのサイズは 1 MB でも同じサイズになります。4 つの OSD が同時に失敗しても、オブジェクトは失われません。

プロファイルで最も重要なパラメーターは、ストレージのオーバーヘッドとデータの永続性を定義するため、**k**、**m**、および **crush-failure-domain** です。

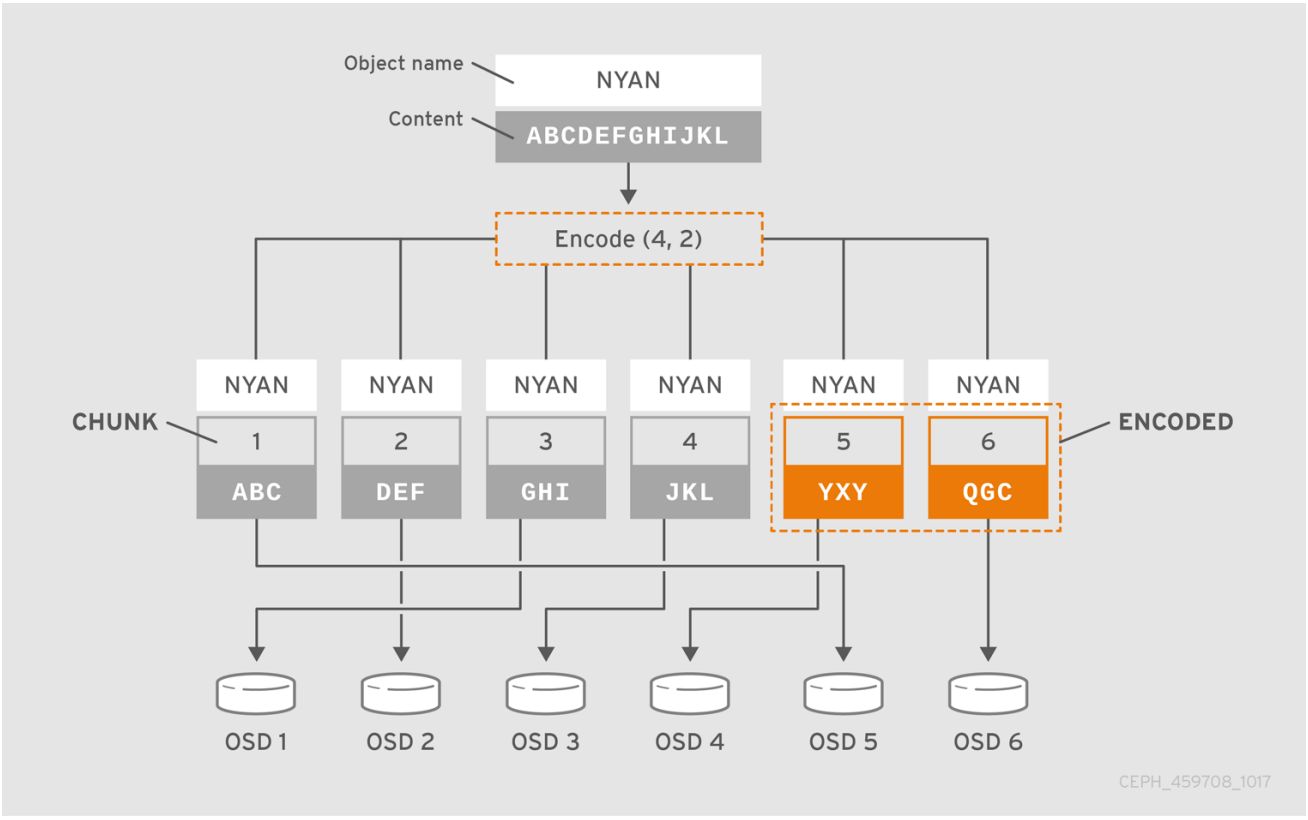
重要

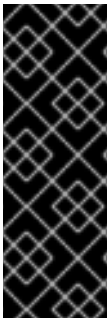
プールの作成後にプロファイルを変更できないため、正しいプロファイルの選択は重要です。プロファイルを変更するには、別のプロファイルで新しいプールを作成し、オブジェクトを古いプールから新しいプールに移行する必要があります。

たとえば、目的のアーキテクチャーがストレージオーバーヘッドの 40% オーバーヘッドの 2 つのラックが失われなければならない場合、以下のプロファイルを定義することができます。

```
$ ceph osd erasure-code-profile set myprofile \
   k=4 \
   m=2 \
   crush-failure-domain=rack
$ ceph osd pool create ecpool 12 12 erasure *myprofile*
$ echo ABCDEFGHIJKL | rados --pool ecpool put NYAN -
$ rados --pool ecpool get NYAN -
ABCDEFGHIJKL
```

プライマリー OSD は、**NYAN** オブジェクト を 4 つ (**k=4**) のデータチャンクに分割し、2 つの追加のチャンク (**m=2**) を作成します。**m** の値は、データを失うことなく同時に失われた OSD の数を定義します。**crush-failure-domain=rack** は、2 つのチャンクが同じラックに保存されないように CRUSH ルールを作成します。

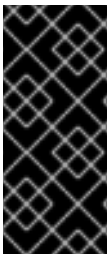




重要

Red Hat は、**k** および **m** に以下の **jerasure** コーディング値をサポートしています。

- k=8 m=3
- k=8 m=4
- k=4 m=2



重要

失われた OSD の数がコーディングチャンクの数 (**m**) と同等である場合、イレイジャーコーディングプールの一部の配置グループは不完全な状態になります。失われた OSD の数が **m** 未満の場合、配置グループは不完全な状態になりません。いずれの場合も、データの損失は発生しません。配置グループが不完全な状態の場合は、イレイジャーコード化されたプールの **min_size** を一時的に縮小することで復元が可能になります。

5.2.1. OSD の erasure-code-profile の設定

新しい元号のコードプロファイルを作成するには、以下の手順を実施します。

構文

```
ceph osd erasure-code-profile set NAME \
     [<directory=DIRECTORY>] \
     [<plugin=PLUGIN>] \
     [<stripe_unit=STRIPE_UNIT>] \
     [<_CRUSH_DEVICE_CLASS_>]\
     [<_CRUSH_FAILURE_DOMAIN_>]\
     [<key=value> ...] \
     [--force]
```

ここでは、以下のようになります。

directory

説明

イレイジャーコードプラグインが読み込まれた **ディレクトリー** 名を設定します。

型

String

必須

いいえ

デフォルト

/usr/lib/ceph/erasure-code

プラグイン

説明

イレイジャーコードプラグインを使用して、コーディングしたチャンクを計算して、欠落しているチャンクを回復します。詳細は、イレイジャーコードプラグイン セクションを参照してください。

型

String

必須

いいえ

デフォルト

jerasure

stripe_unit

説明

ストライプごとにデータチャンクのデータの量。たとえば、2 つのデータチャンクと **stripe_unit=4K** のプロファイルでは、範囲 0 ~ 4,000 がチャンク 0 に設定され、4,000 ~ 8,000 がチャンク 1 に、8,000 ~ 12,000 がチャンク 0 に再び設定されます。最高のパフォーマンスを得るためには、4K の倍数である必要があります。デフォルト値は、プールの作成時に監視設定オプション **osd_pool_erasure_code_stripe_unit** から取得されます。このプロファイルを使用するプールの stripe_width は、この **stripe_unit** で乗算したデータチャンクの数になります。

型

String

必須

いいえ

デフォルト

4K

crush-device-class

説明

hdd、**ssd** などのデバイスクラス。

型

String

必須

いいえ

デフォルト

none (CRUSH がクラスに関係なくすべてのデバイスを使用することを意味します)。

crush-failure-domain

説明

host、**rack** などの障害ドメイン。

型

String

必須

いいえ

デフォルト

host

key

説明

残りのキーと値のペアのセマンティックは、イレイジャーコードプラグインによって定義されます。

型

String

必須

いいえ

--force

説明

同じ名前で既存のプロファイルを上書きします。

型

String

必須

いいえ

5.2.2. OSD の erasure-code-profile の削除

イレイジャーコードプロファイルを削除するには、以下のコマンドを実行します。

構文

```
ceph osd erasure-code-profile rm NAME
```

プロファイルがプールで参照される場合、削除は失敗します。

5.2.3. OSD の erasure-code-profile の取得

イレイジャーコードプロファイルを表示するには、以下のコマンドを実行します。

構文

```
ceph osd erasure-code-profile get NAME
```

5.2.4. OSD の erasure-code-profile の一覧表示

イレイジャーコードプロファイルの名前をリスト表示するには、以下のコマンドを実行します。

構文

```
ceph osd erasure-code-profile ls
```

5.3. 上書きによるイレイジャーコーディング

デフォルトでは、イレイジャーコーディングのプールは Ceph Object Gateway でのみ機能します。これにより、全オブジェクト書き込みを実行し、追記します。

上書きによるイレイジャーコードプールを使用すると、Ceph のブロックデバイスと CephFS は、イレイジャーコードプールにデータを保存できます。

構文

```
ceph osd pool set ERASURE_CODED_POOL_NAME allow_ec_overwrites true
```

例

```
[ceph: root@host01 /]# ceph osd pool set ec_pool allow_ec_overwrites true
```

上書きによるイレイジャーコードプールを有効にすることで、BlueStore OSD を使用するプールにしか存在できません。BlueStore のチェックサムは、ディレップスクラブ中にビットロットやその他の破損を検出するために使用されます。

イレイジャーコードプールは omap をサポートしません。Ceph Block Devices および CephFS で元号のコードプールを使用するには、イレイジャーコードプールにデータを、複製プールにメタデータを保存します。

Ceph ブロックデバイスの場合は、イメージの作成時に **--data-pool** オプションを使用します。

構文

```
rbd create --size IMAGE_SIZE_M|G|T --data-pool _ERASURE_CODED_POOL_NAME
REPLICATED_POOL_NAME/IMAGE_NAME
```

例

```
[ceph: root@host01 /]# rbd create --size 1G --data-pool ec_pool rep_pool/image01
```

CephFS にイレイジャーコードプールを使用している場合には、ファイルレイアウトで上書きを設定する必要があります。

5.4. イレイジコードプラグイン

Ceph では、プラグインアーキテクチャーとのイレイジャーコーディングがサポートされます。つまり、さまざまなタイプのアルゴリズムを使用して、イレイジャーコードプールを作成できます。Ceph は Jerasure をサポートしています。

5.4.1. jerasure イレイジャーコードプラグインを使用した新しいイレイジャーコードプロファイルの作成

jerasure プラグインは、最も汎用的で柔軟性のあるプラグインです。Ceph Erasure コードプールのデフォルトでもあります。

jerasure プラグインは JerasureH ライブラリーをカプセル化します。パラメーターの詳細は、**jerasure** のドキュメントを参照してください。

jerasure プラグインを使用して新しいイレイジャーコードプロファイルを作成するには、以下のコマンドを実行します。

構文

```
ceph osd erasure-code-profile set NAME \
    plugin=jerasure \
    k=DATA_CHUNKS \
```

```
    m=DATA_CHUNKS \
    technique=TECHNIQUE \
    [crush-root=ROOT] \
    [crush-failure-domain=BUCKET_TYPE] \
    [directory=DIRECTORY] \
    [--force]
```

ここでは、以下のようになります。

k

説明

各オブジェクトは **data-chunks** の部分で分割され、それぞれが異なる OSD に保管されます。

型

整数

必須

必須です。

例

4

m

説明

各オブジェクトの **コーディングチャンク** を計算し、それらを異なる OSD に保存します。コーディングのチャンクの数は、データが失われることなくダウンできる OSD 数でもあります。

型

整数

必須

必須です。

例

2

テクニック

説明

より柔軟な技術は **reed_sol_van** で、**k** と **m** を設定するだけで十分です。**cauchy_good** 技術は速くなりますが、慎重に **packetsize** を選択する必要があります。**reed_sol_r6_op**、**liberation**、**blaum_roth**、**liber8tion** はすべて、**m=2** でしか設定できない意味で **RAID6** と同等です。

型

String

必須

いいえ

有効の設定

reed_sol_vanreed_sol_r6_opcauchy_origcauchy_goodliberationblaum_rothliber8tion

デフォルト

reed_sol_van

packetsize

説明

エンコーディングは、**バイト** サイズのパケットで一度に行われます。適切なパケットサイズの選択は困難です。**jerasure** ドキュメントには、このトピックに関する詳細な情報が記載されています。

型

整数

必須

いいえ

デフォルト

2048

crush-root

説明

ルールの最初のステップに使用される CRUSH バケットの名前。たとえば、**step take default** となります。

型

String

必須

いいえ

デフォルト

default

crush-failure-domain

説明

同じ障害ドメインを持つバケットに 2 つのチャンクがないことを確認します。たとえば、障害ドメインが **ホスト** の場合、2 つのチャンクは同じホストに保存されません。これは、**step chooseleaf host** などのルールステップを作成するのに使用します。

型

String

必須

いいえ

デフォルト

host

directory

説明

イレイジャーコードプラグインが読み込まれた **ディレクトリー** 名を設定します。

型

String

必須

いいえ

デフォルト

/usr/lib/ceph/erasure-code

--force

説明

同じ名前で既存のプロファイルを上書きします。

型

String

必須

いいえ

5.4.2. CRUSH 配置の制御

デフォルトの CRUSH ルールは、異なるホストにある OSD を提供します。以下に例を示します。

```
chunk nr    01234567

step 1      _cDD_cDD
step 2      cDDD____
step 3      ____cDDD
```

各チャンクに 1 つずつ、正確に 8 つの OSD が必要です。ホストが 2 つ隣り合ったラックにある場合は、最初の 4 つのチャンクを最初のラックに配置し、残りを 2 番目のラックに配置できます。1 つの OSD の失われた状態からのリカバリーには、2 つのラック間で帯域幅を使用する必要はありません。

以下に例を示します。

```
crush-steps='[ [ "choose", "rack", 2 ], [ "chooseleaf", "host", 4 ] ]'
```

rack タイプの CRUSH バケットを 2 つ選択し、それぞれに 4 つの OSD (**host** タイプの異なるバケットに配置) を選択するルールを作成します。

また、細かい制御のためにルールを手動で作成することもできます。