リバランスの理解 パート 1: リバランス実行中の処理内容

February, 2018

原文はこちら

MICRO FOCUS

目次

リバランスとは

Vertica クラスターで 1 つ以上のノードを追加または削除した後、既存および新規ノードのデータを調整する必要があります。最適なパフォーマンスを得るには、すべてのノード上でデータが均一に配置されている必要があります。Vertica では、この処理を*リバランス*と呼びます。

クラスターのリバランスを行った後、データストレージとワークロードはクラスター内のすべてのノード上で均一な状態となります。リバランスは複雑な処理であり、CPU、ディスク、ネットワークを大量に使用します。リバランスには大量のデータ移動が必要なため、処理に時間がかかることがあります。

本書は、2 つのパートからなるリバランスについてのシリーズのパート 1 です。第 1 部では、リバランス実行中に何が起こるかを説明します。第 2 部（Understanding Rebalancing, Part 2: Optimizing for Rebalancing）では、次のような、リバランスの前、中、後に取るべき手順について説明します。

- リバランスの準備
- リバランス実行中のモニタリング
- リバランス結果の確認

クラスター内にノードを追加または削除するケース

次の場合、Vertica クラスターに 1 つ以上のノードを追加する必要がありえます。

- データベースのデータ量が大幅に増加した場合。データ用にディスクストレージが不足している場合、パフォーマンスが低下する可能性があります。
- データベースの分析ワークロードが大幅に増加した場合。この状況により、パフォーマンスが低下する可能性があります。
- 高可用性を確保するには、クラスター内の K-Safety を増やす必要がある場合
- メンテナンス、アップグレード、または交換のためにノードをクラスターから交換する必要がある場合。ノードを交換する場合、Vertica はノード間でデータをリバランスする必要はありません。

ノードの削除は、ノードの追加よりも一般的ではありません。クラスターが過剰にプロビジョニングされている場合、または別の目的でハードウェアを転用する必要がある場合、ノードを削除する可能性があるでしょう。

重要

ノードを削除するとシステムの K-safety に反する場合、Vertica はクラスターからノードを削除できません。

リバランス実行中の処理内容

次のトピックでは、ノードを追加または削除した後にクラスターをリバランスする際のの処理内容について説明します。

- リバランシング中のデータ移動
- REFRESH リソースプール
- リバランスのフェーズ

リバランス中のデータ移動

クラスター全体でのデータのリバランスは複雑です。Vertica は、適切なセグメントをそれぞれの宛先ノードに転送する前に、分散されたプロジェクションを分割します。リバランスが完了した後、宛先ノード上で、Tuple Mover は次回実行時にデータセグメントをマージします。

効率的なリバランスのために、既存のノードには空き領域が必要です。既存のノードに空き領域がほとんどない場合でも、リバランスは実行可能でしょう。しかしながら、Vertica は複数の段階をもってリバランスを実行しなければならくなり、これにはかなりの時間を要します。

1. 第 1 段階で、Vertica は新しいノードにデータを分配します。
2. 次の段階で、Vertica はデータを新しいノードに送信したノードにデータを分配します。
3. 後続の段階では、Vertica は前の段階で分配されたデータをオフロードすることによってスペースを解放したノードにデータを送信し続けます。

たとえば、N 個のノードを持つクラスターを考えてみましょう。クラスターにノードを追加すると、Vertica はすべてのノードにデータを均等に分散させるために必要なデータ移動量を最小限におさえようとします。このため、Vertica は既存のノードの間に新しいノードを分配します。この点については、このセクションの後の図を参照してください。

リバランス中の Vertica 上のデータ移動量は、次の項目によって決まります。

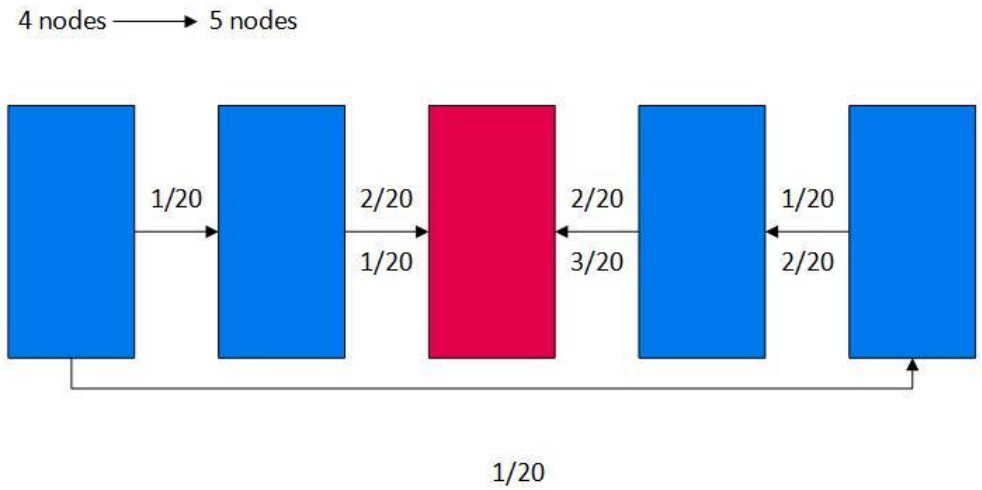
- 既存クラスターのノードの数
- 追加するノードの数
- 分散化されていないプロジェクションと分散化されたプロジェクションの数。たとえば、Vertica は、各ノードにデータの完全コピーが格納されているため、バディノードから分散化されていないプロジェクションをコピーします。

次の図は、このプロセスがプライマリーとバディーの両方のプロジェクションに対してどのように実施されるかを示しています。青色の四角形は既存のノードを表し、緑色の四角形は新しいノードを表します。

- Vertica は、データ移動を最小限に抑える位置で、ノードをクラスターに挿入します。
- Vertica は、新しいノードと既存のノードの両方にデータを転送します。グラフィックの矢印はデータ転送の方向と移動した割合を示します。

たとえば、グラフィックの一番上の行は、1 つのノードを 4 ノードクラスターに追加することを示します。Vertica は、データ移動を最小限にする場所に新しいノードを分配します。

4 ノードのクラスターでは、各ノードにはデータの 1/4 が含まれていました。5 ノードのクラスターでは、各ノードに 1/5 のデータが含まれている必要があります。クラスターが 4 ノードから 8 ノードに倍増する場合も、同じ概念が適用されます。



REFRESH リソースプール

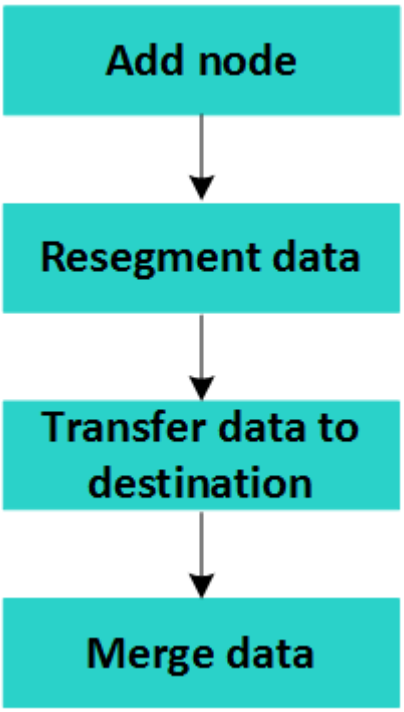
リバランスは、ビルトインの REFRESH リソースプールを使用して実行されます。このプールでは、PLANNEDCONCURRENCY パラメータを使用して、いつでも Vertica がリバランスできるプロジェクションのバディーグループの数を指定できます。MAXCONCURRENCY パラメーターは、REFRESH リソースプールに影響しません。

REFRESH リソースプールは、デフォルト設定を使用しましょう。

リバランスのフェーズ

大量のデータ移動があるため、ディスクスペースを節約するために、Vertica は一度にテーブルのグループとプロジェクションのグループをリバランスします。グループの数は、PLANNEDCONCURRENCY パラメーターの値によって異なります。

リバランスのフェーズは次の通りです。



データをマージする最後のフェーズは、リバランス中には発生しません。Tuple Mover は、次にマージアウトを実行するときにデータをマージします。

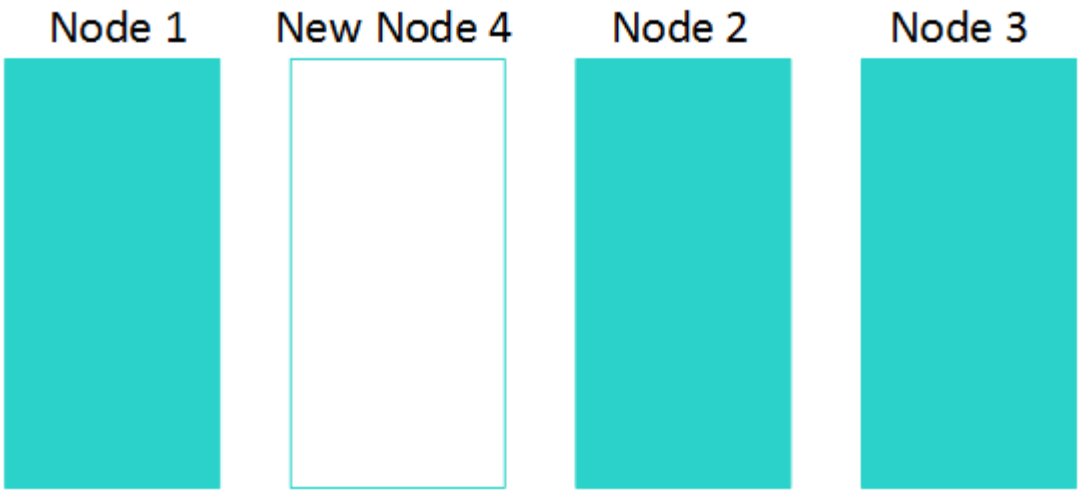
詳細については、以下のトピックを参照してください。

- Adding a Node
- Resegmenting the Data
- Transferring the Data to Destination Nodes
- Merging the Data

ノードの追加

クラスターのノード間でデータを移動するのにかかる時間を最小限に抑えるために、Vertica はデータ移動を最小限にする場所に新しいノードをクラスターに挿入します。新しいノードの位置は、リバランスのパフォーマンスに影響します。これは、大規模なクラスターの場合に特に当てはまります。

3 ノードのクラスターに 1 つのノードを追加すると、次のようになります。



データの再分散

Vertica はすべてのノードから既存のデータを読み取り、各テーブルとプロジェクションを見ます。

分散化されていないプロジェクションの場合、Vertica は次のように処理します。

- 各プロジェクションに対して X ロックを取得
- 次のコマンドを使用して、宛先ノードのプロジェクションを複製

```
=> CREATE PROJECTION ... UNSEGMENTED ALL NODES KSAFE
```

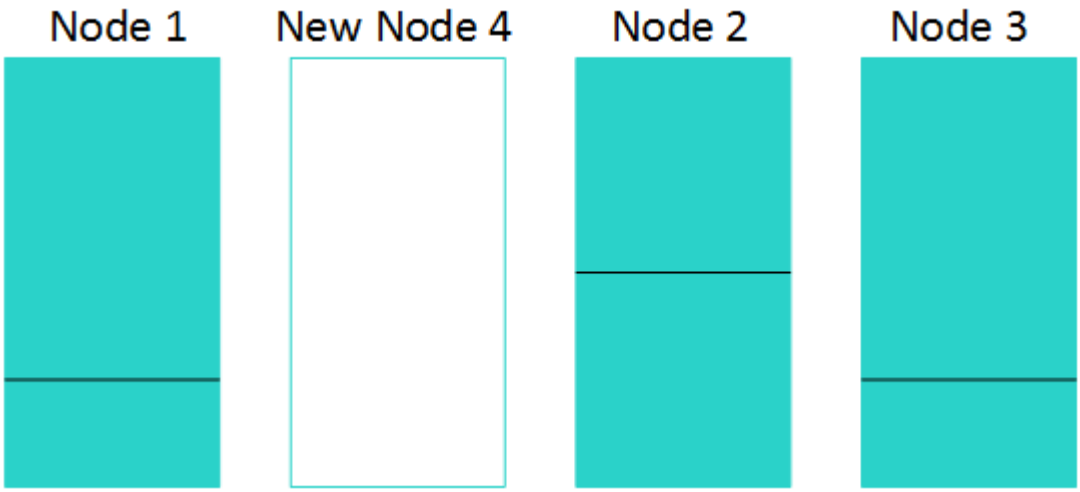
- バディープロジェクションからプロジェクションをリフレッシュ

分散化されたプロジェクションの場合、Vertica は次のように処理します。

1. テーブルに対して S ロックを取得し、プロジェクションに対して X ロックを取得
2. プライマリー、バディー、およびライブアグリゲートプロジェクションのセグメントを分割
3. プロジェクションをリフレッシュ

データの分散化にはステージング領域が必要なため、リバランスは一時記憶域を使用します。ストレージを効率的に使用するために、Vertica は一度に少数のテーブルとプロジェクションだけをリバランスします。

3 ノードクラスターにノードを追加した後、再分散は次のようになります。



宛先ノードへのデータ転送

新しいサイズのクラスターのバランスをとるために、Vertica はハッシュ関数を使用して、新しいノードと既存のノードにデータをどのように配布するかを決定します。データを転送する際、Vertica は S ロックを取得し、分散化されていないデータと分散化されたデータをコピーします。

分散化されていないプロジェクションは双方のコピーであるため、ソースノードはデータを読み取り、宛先ノードはデータを書き込みます。複数の新しいノードがある場合、Vertica は、複数のソースノードから、分散化されていないプロジェクションを複数の宛先ノードに並行して転送することができます。このプロセスでは CPU コストがほとんどかかりません。

分散化されたプロジェクションの場合、これらのステップはより複雑です。ソースノードでは、Vertica は分散化されたプロジェクションを読み込み、分割して書き出します。Vertica は、これらの操作を実行するために時間とディスクスペースを必要とします。

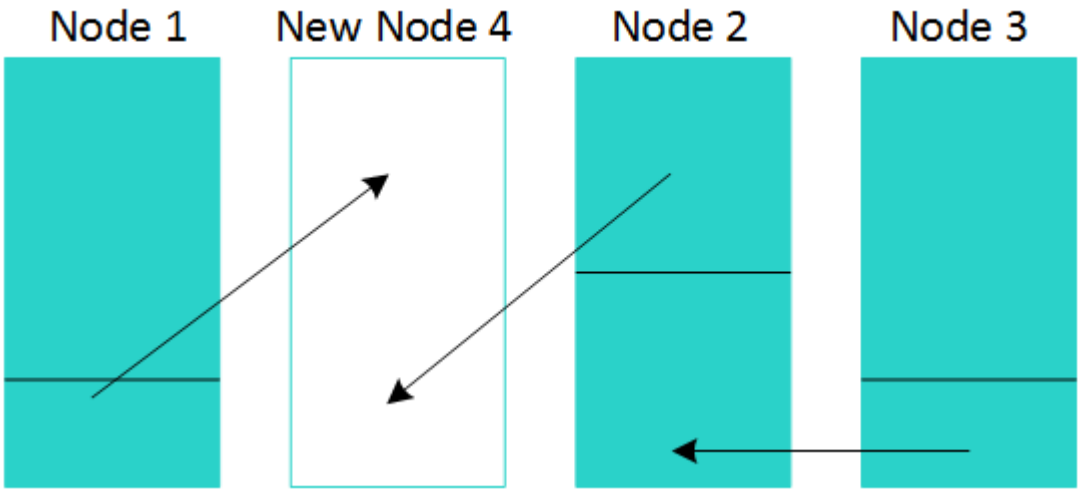
リバランスが完了した後、ターゲットノード上で、最終的に Tuple Mover がデータセグメントをマージします。

3 ノードの例では、Vertica が新しいノードに隣接ノードからのデータを演奏して、データ転送の量を最小限に抑えることがわかります。転送後、データは 4 つのノードすべてで均衡な状態となります。

この例では、次のようになります。

- ノード 1 は、データベース内の全データの 1/12 をノード 4 に転送します。
- ノード 2 は、データベース内の全データの 2/12 をノード 4 に転送します。
- ノード 3 は、データベース内の全データの 1/12 をノード 2 に転送します。

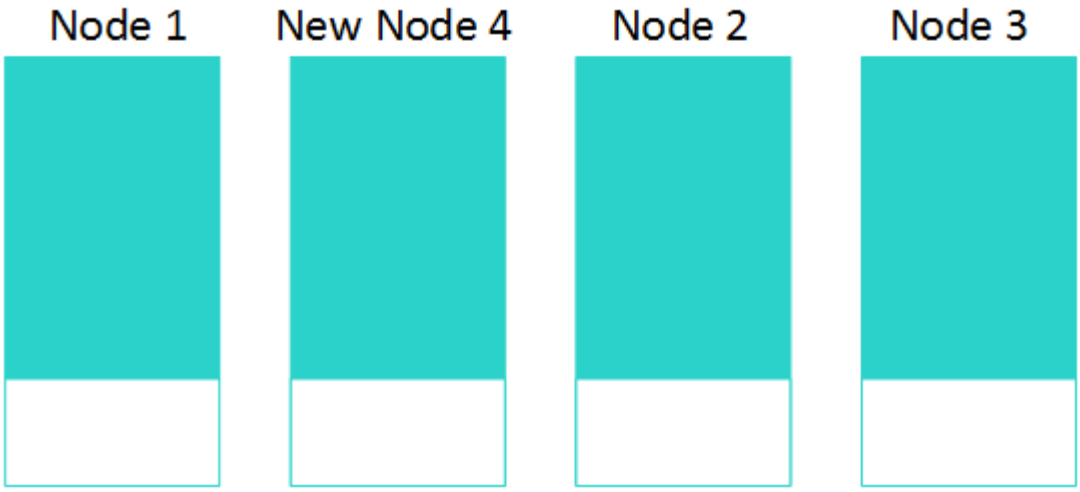
この結果、すべてのノードが Vertica データベースの全データの 1/4 を保持するようになります。



データのマージ

リバランスが完了した後、宛先ノードで、Tuple Mover は次のマージアウト操作中にデータセグメントをマージします。

次の図は、4 番目のノードを追加した後の 3 ノードクラスターの状況を示しています。



Tuple Mover はデータをマージした後、すべてのプロジェクションをリフレッシュします。Ephemeral ノードでノードを削除する場合、Vertica は不要な分散化されていないプロジェクションを削除します。

リバランシングにかかる時間

クラスターのリバランスには長い時間がかかることが考えられます。以下の要因のいずれかが、リバランスが完了するまでの時間に影響する可能性があります。

- プロジェクション数
- テーブルごとのパーティション数
- データ量とプロジェクション内の行数
- 宛先ノード上でデータをマージするためにかかる時間
- 最もデータのやりとりの多いノードの合計データ移動量(読み取りと書き込み)
- データスキュー
- ネットワークスループット
- リバランス処理が I/O バウンドまたはネットワークバウンドの場合
- クラスター上のその他のワークロード

分散化されたプロジェクションを再分散化し、ROS コンテナを分割することは、リバランスの合計時間の80%を要する可能性があります。

ノードを追加または削除する前に

クラスター内のノードを追加または削除する前に、次の手順を実行してリバランスのパフォーマンスを最適化し、データ損失のリスクを最小限に抑えてください。

1. データベースをバックアップします。
2. 古いテーブルパーティションまたは未使用のパーティションを削除します。
3. デフォルトの設定の通り、ローカルセグメンテーションが無効になっていることを確認します。リバランスを開始する前に、ローカルセグメンテーションを無効にする必要があります。ローカルセグメンテーションを無効にするには、次のコマンドを使用します。

   ```
   => SELECT DISABLE_LOCAL_SEGMENTS();
   ```
4. リバランス処理を実行するために使用できる CPU およびネットワーク帯域幅の量を確認します。これを行うには、次の Vertica ツールを使用します。
   - vioperf: ハードドライブの速度と一貫性を測定
   - vnetperf: ノード間のネットワークのレイテンシーとスループットを測定
5. リバランスを実行するのに十分なディスクスペース(データベースのサイズの少なくとも 40%)があるかどうかを確認します。ノード間でデータを移動する場合、リバランス処理では、中間処理に大量のディスク領域が使用されます。

   十分な空き容量がない場合、Vertica は複数の段階でリバランスを実行する必要がでてきますが、これにはより時間がかかります。

   空きディスク容量については、次のシステムテーブルを確認してください。
   - DISK_STORAGE—各ノード上でデータベースが使用するディスクストレージの量

- COLUMN_STORAGE—各プロジェクションの各列が各ノード上で使用するディスクストレージの量
- PROJECTION_STORAGE—各プロジェクションが各ノード上で使用するディスクストレージの量

Linux ファイルシステム上で使用可能なディスクスペースと使用済みディスクスペースを確認するには、Linux の df コマンドを使用します。

```
$ df -h
```

各ノードのスナップショットを取得するには、HOST_RESOURCES システムテーブルの次のフィールドを確認します。

```
=> SELECT host_name, disk_space_used_mb, disk_space_total_mb FROM
host_resources;
```

6. ビルトインの REFRESH リソースプールの設定を確認するには、次のステートメントを入力します。必要に応じて、設定を調整します。

```
=> SELECT name, is_internal, plannedconcurrency, maxmemorysize FROM
resource_pools WHERE name='REFRESH';
```

7. リバランスされるテーブルの DML 操作(COPY、INSERT、UPDATE、DELETE)を最小限に抑えます。リバランスがテーブルのロックを取得している場合、ロードは失敗します。ロードがテーブルをロックしている場合、リバランスは一時停止します。

8. Purging Deleted Data の説明に従って、論理削除されたデータを物理削除します。

9. Connection Load Balancing の手順に従って、クラスターに追加するホストを設定します。

10. Adding Hosts to a Cluster に記載のある手順を使用して、ホストをクラスターに追加します。

11. Adding Nodes to a Database の記載に従って、ノードをデータベースに追加します。

詳細情報

Understanding Rebalancing, Part 2: Optimizing for Rebalancing を参照してください。

Vertica 製品のドキュメント内ののリバランシング関連情報については、Rebalancing Data Across Nodes を参照してください。