Tuple Mover に関するベストプラクティス

Applies to Vertica 7.2.x and earlier

March, 2017

原文はこちら

MICRO FOCUS

目次

Tuple Mover の概要 3
Tuple Mover Moveout 処理 3
WOS Spillover（スピルオーバー）の検出 3
Moveout に関するベストプラクティス 3
大容量ファイルのロードには COPY DIRECT を使用する 4
構成パラメーター: MoveOutInterval 4
コミットされていない WOS 上のデータ 4
大きな一時テーブルには WOS を使用しない 4
WOSDATA リソースプールの maxMemorySize 4
構成パラメーター: MoveOutSizePct 4
構成パラメーター: MoveOutMaxAgeTime 4
Moveout に関するよくあるご質問 5
Moveout 処理のスレッド数を増やすことは可能でしょうか？ 5
MoveOutInterval パラメーターの変更方法と適用タイミングは？ 5
各ノード上での moveout の発生状況を確認する方法は？ 5
Moveout が取得するロックは？ 5
Moveout はどのようなときに失敗しますか？ 5
Tuple Mover Mergeout 処理 5
Mergeout に関するベストプラクティス 5
WOS を使用するトリクルロードか、DIRECT ヒント付のバッチロードか 6
TM リソースプールの MemorySize 6
テーブルパーティション 6
構成パラメーター: ActivePartitionCount 6
同一テーブルにひもづけられたプロジェクションセットを制限する 6
プロジェクションのソート順に関するガイドライン 6
Delete や Replay Delete のためのプロジェクション最適化 6
再編成処理が正常に完了したかどうかの確認方法 7
可能な限り、Delete や Update は一括実行する 7
構成パラメーター: MergeOutInterval 7
TM リソースプールの MaxConcurrency と PlannedConcurrency 7
MergeOutCache による Mergeout の効率化 7
Mergeout に関するよくあるご質問 7
Tuple Mover mergeout の STRATA アルゴリズムはどのように動作するのでしょうか？ 7
非アクティブパーティション上で mergeout のスレッドはいくつ動作するのでしょうか？ 9
Mergeout 処理は論理削除済みデータの物理削除も実行するのでしょうか？ 9
以下の構成パラメーターはいつどのように変更できるのでしょうか？ 9
プロジェクションのアクティブパーティションをどのように確認できるのでしょうか？ 9
ノードごとプロジェクションごとの ROS コンテナの数をどのように確認できるのでしょうか？ 9

Tuple Mover の概要

Vertica analytics platform は、2 つのストレージオプションを提供しています。1 つは小さなデータファイルをトリクルロードするための WOS と呼ばれるメモリー領域、もう 1 つは大きなデータファイルをバルクロードするための ROS と呼ばれるファイルシステムです。WOS にロードされるデータは、ソートされていないデータとして格納されますが、ROS にロードされるデータは、プロジェクションデザインに基づいてソートされ、エンコード・圧縮されたデータとして格納されます。

Tuple Mover は、バックグラウンドで実行される Vertica のサービスで、以下の 2 種類の処理を含みます。

- **Moveout**：Tuple Mover の moveout 処理は、WOS コンテナから新しい ROS コンテナにデータを定期的に移動し、WOS がいっぱいになって ROS にスピルする（流出する）ことを防ぎます。Moveout は、WOS コンテナの特定のセット上で、一度に 1 つのプロジェクションを対象に実行されます。Moveout 操作がプロジェクションを選択して ROS にデータを移動する際、それまでにコミットされたすべてのトランザクションからロードされたプロジェクションのデータを結合し、単一の ROS コンテナに書き込みます。
- **Mergeout**：Tuple Mover の mergeout 処理は、ROS コンテナを統合し、Delete によって論理削除されたレコードを物理的に削除します。

ほとんどのユースケースでは、Tuple Mover はデフォルトのパラメーターを超えた構成をほとんど必要としません。ただし、一部のワークロードでは、構成パラメーターのチューニングが必要な場合があります。
このドキュメントでは、

- よくあるご質問にお答えします
- トラブルシューティングの Tips をご提供します
- WOS、データロード、およびスキーマ設計に関するベストプラクティスについてご説明します

本書は、Tuple Mover 処理およびシステム全体の効率を向上させることを目的に記載されています。

Tuple Mover Moveout 処理

WOS Spillover（スピルオーバー）の検出

WOS 用のメモリーは、WOSDATA という名前のビルトインのリソースプールによって制御されます。
WOSDATA のデフォルトの最大メモリサイズはノードあたり 2GB です。Tuple Mover がデータを移動する速度よりも速く WOS にデータをロードすると、データは WOS の領域が使用可能になるまで ROS にスピルする（流出する）可能性があります。このスピルオーバーでは データ損失は発生しませんが、ROS コンテナが予想よりもはるかに高速に作成され、moveout 処理が遅延してしまいます。
次のクエリを実行すると、データベース上で WOS のスピルオーバーが発生しているかどうかを検出できます。

```
=>SELECT node_name, count(*) from dc_execution_engine_events
 WHERE event_type = 'WOS_SPILL'
 group by node_name;
```

Moveout に関するベストプラクティス

次のベストプラクティスに従って、Tuple Mover の moveout 処理が円滑に正しく実行されることを確認します。

大容量ファイルのロードには COPY DIRECT を使用する

大きなデータファイル（ノードあたり 100MB 以上）をロードする場合、大きな COPY 文を COPY DIRECT 文に変更します。これにより、WOS へのロードをバイパスし、ファイルを ROS に直接ロードすることができます。

構成パラメーター: MoveOutInterval

このパラメーターには、WOS の半分を満たすのにかかる時間よりも短い値を設定します。このパラメーターは、Tuple Mover の moveout 処理が WOS から移動するものがないことを検出したときにスリープする時間を秒単位で指定します。このパラメーターのデフォルト値は 300 秒です。この値を小さくすると、WOS から ROS にデータをより頻繁に移動できます。

コミットされていない WOS 上のデータ

Tuple Mover はコミットされたデータのみを移動するため、コミットされていないデータを必要以上に WOS に残し続けないでください。WOS がコミットされていないトランザクションからのデータで満たされている場合、moveout はデータを移動できません。これにより、ROS への WOS スピルオーバーが発生する可能性があります。

大きな一時テーブルには WOS を使用しない

一時テーブルに対して、大きなデータセット（ノードあたり 50 MB 以上）をロードする際に WOS を使用しないでください。Moveout 処理は、一時テーブルのデータを移動しません。データは、トランザクションまたはセッションが終了すると破棄されます。

WOSDATA リソースプールの maxMemorySize

同時に 10 以上のテーブルにトリクルロードを実行し、WOS のスピルオーバーが発生した場合は、これまでのベストプラクティスが問題の解決に役立つはずです。WOSDATA リソースプールの maxMemorySize を大きくすると、WOS のメモリーサイズを大きくすることができます。**ただし、積極的に多くのテーブルにデータをロードし、プロジェクションあたりの WOS のデータが数百メガバイトを超えない限り、WOS のメモリーサイズを大きくしないでください。**プロジェクションごとに WOS に数百 MB 以上のデータがある場合、クエリパフォーマンスにばらつきが発生する可能性があります。

構成パラメーター: MoveOutSizePct

WOS の使用率に基準を設定して moveout のトリガーとするために、このパラメーターを設定します。このパラメータを設定すると、データを少数のテーブルにロードする場合に役立ちます。このような場合、moveout 処理は積極的に WOS からデータを移動し、頻繁に小さな ROS コンテナを作成することがあります。たとえば、この値を 40 に設定した場合、WOS が 40%いっぱいになるまでプロジェクションを moveout せず、まとまった単位で moveout することができます。

構成パラメーター: MoveOutMaxAgeTime

経過時間（秒）に従って、Tuple Mover が moveout を実行する前に、WOS にデータが存在する時間を設定するには、このパラメーターを設定します。このパラメーターのデフォルト値は 1800 です。上述の MoveOutSizePct パラメーターの値を変更する場合は、このパラメーターの値を 600 に変更できます。

Moveout に関するよくあるご質問

Moveout 処理のスレッド数を増やすことは可能でしょうか？

いいえ。Moveout のスレッドのデフォルト数は 1 で、この値を変更することはできません。

MoveOutInterval パラメーターの変更方法と適用タイミングは？

この構成パラメーターは、vsql で set_config_parameter コマンドを実行することによって変更できます。このパラメーターを変更すると、moveout のスリープ状態に影響を及ぼします。次の例では、このパラメーターの値を 120 秒に変更します。

```
=>SELECT set_config_parameter('MoveOutInterval',120);
```

各ノード上での moveout の発生状況を確認する方法は？

次の SQL 文を実行すると、Tuple Mover 処理の進行状況を確認できます。

```
=>SELECT * from tuple_mover_operations where is_executing;
```

Moveout が取得するロックは？

プロジェクションのデータを移動するとき、moveout はテーブルに対して U ロックを取得します。Delete または replay delete の処理を反映する際には、moveout はテーブルに対して T ロックをとります。

Moveout はどのようなときに失敗しますか？

- Moveout 処理が T ロックを取得できない場合、処理が失敗する、またはブロックされます。これは、同じテーブルに X ロックを保持しているトランザクションが長時間実行中であるために発生します。
- 長時間実行されている mergeout 処理の対象となっている ROS コンテナに関連するデリートベクターを moveout しようとしている場合、処理は失敗する、またはブロックされます。
- WOS に 1024 を超えるテーブルパーティション分のデータをロードした場合も、moveout 処理に失敗します。

Tuple Mover Mergeout 処理

Tuple Mover mergeout 処理は、バックグラウンドで実行され、ROS コンテナを統合する Vertica サービスです。ノードあたりプロジェクションあたりの ROS コンテナの最大数は 1024 です。データロード中に「TOO MANY ROS CONTAINERS」というエラーが発生した場合は、ノードあたりプロジェクションあたりの最大 ROS コンテナ数に達しています。Mergeout は、ROS コンテナをより少ないコンテナに統合し、通常の使用状況ではデータベースがこの制限に達しないことを確認します。

Mergeout に関するベストプラクティス

Tuple Mover の mergeout 処理が円滑に実行されていることを確認し、ROS コンテナの蓄積を減らすために、以下のベストプラクティスに従ってください。

WOS を使用するトリクルロードか、DIRECT ヒント付のバッチロードか

WOS へのトリクルロードは、ディスクに移動する前にデータをまとめるため、生成される ROS コンテナの数を減らすことに寄与します。大容量ファイルの場合は、DIRECT ヒントを使用して ROS にバッチロードするようにします。

TM リソースプールの MemorySize

データベースに列数の多いテーブル（100 列以上）がある場合は、TM リソースプールの MemorySize をデフォルトの 200MB から 6GB に増やし、PLANNEDCONCURRENCY パラメーターを 3 に変更します。Tuple Mover の mergeout スレッドごとに少なくとも 2GB を確保することで、列数の多いテーブルに対する mergeout 処理を高速化することができます。

テーブルパーティション

Vertica はパーティション間で ROS コンテナをマージしないため、テーブルごとに 50 個以下のパーティションを作成するようにします。テーブルが数百のパーティションを持つ場合は、早期に ROS プッシュバックが発生してしまう可能性があります。ALTER TABLE ALTER PARTITION コマンドを使用して、テーブルのパーティションスキームを変更できます。

重要:アクセスしなくなった古いパーティションがある場合は、MOVE_PARTITION_TO_TABLE 関数を使用して、それらをアーカイブテーブルに移動できます。

構成パラメーター: ActivePartitionCount

テーブルが、現在のアクティブパーティションそして最新の非アクティブパーティションにデータを頻繁に受け取る場合は、ActivePartitionCount パラメーターをデフォルト値の 1 から 2 に変更します。Verica は、パーティションは時間ベースであり、1 つのアクティブパーティションがデータを受け、その他の非アクティブパーティションではデータを受けることがほとんどないことを期待しています。Tuple Mover は、非アクティブパーティション用の ROS コンテナを単一の ROS コンテナにマージします。

同一テーブルにひもづけられたプロジェクションセットを制限する

同じテーブルに 2 つより多いプロジェクションセットを持たせないでください。1 つのテーブルにつき 2 セットより多いプロジェクションが存在していると、システムリソースの消費につながる可能性があります。

プロジェクションのソート順に関するガイドライン

ソートキーに指定する列の数は 10 未満とし、サイズの大きな VARCHAR 列を使用しないようにします。ソートキーに指定する列の数を制御するため、ソートキーに指定した列の数が少ない新しいプロジェクションで既存テーブルのプロジェクションを置き換え、ソートキーにはサイズの大きな VARCHAR 列を使用しないようにします。これにより、mergeout 処理の実行に要する時間を短縮できます。長時間実行される処理があると、mergeout 処理のスレッドがブロックされ、ROS コンテナの数が増加してしまう可能性があります。

Delete や Replay Delete のためのプロジェクション最適化

高いカーディナリティーの列をソートキーの最後の列として使用することによって、プロジェクションの Delete 処理を最適化します。これは、replay delete のために長時間実行される mergeout 処理を回避するのに役立ちます。Mergeout が replay delete を実行するためにスタックしている場合は、close_session（<session_id>）関数を使用して処理をキャンセルします。その後、make_ahm_now 関数を実行して AHM（Advanced History Mark）エポックを進めます。AHM が進んだ後は replay delete 処理は実行されないため、mergeout 処理はより速く実行されます。Replay delete によって mergeout が長期化すると、時間の経過とともに ROS コンテナが増加してしまう可能性があります。

再編成処理が正常に完了したかどうかの確認方法

PARTITION_STATUS システムテーブルを調べることにより、再編成処理の状況を確認することができます。パーティション式が変更されても再編成式が開始または失敗しない場合は、変更前に存在していたテーブルに固定されているプロジェクションの ROS コンテナは、テーブルが再編成されるまで mergeout の対象になりません。この問題を解決するには、次のコマンドを開始します。

```
=>ALTER TABLE <TABLE_NAME> REORGANIZE;
```

可能な限り、Delete や Update は一括実行する

可能な限り、delete や update は一括で実行することを推奨します。頻繁な delete または update を実行する必要がある場合は、WOS を使用して、デリートベクターをより少ない数に統合します。これにより、ROS コンテナの増加につながるプロジェクションごとのデリートベクターの数を抑えることができます。

構成パラメーター: MergeOutInterval

プロジェクションごとに作成される ROS コンテナの数が ROS プッシュバック制限を超えない範囲で、このパラメーターをデフォルト値の 600 から調整します。

TM リソースプールの MaxConcurrency と PlannedConcurrency

前述のベストプラクティスを検討してもなお、追加の mergeout スレッドが必要なワークロードの場合は、TM リソースプールの MaxConcurrency パラメーターと PlannedConcurrency パラメーターを 3 から 4 に増やします。デフォルト値は 3（1 つの moveout と 2 つの mergeout）です。この値は 6 より大きくしないでください。

MergeOutCache による Mergeout の効率化

Vertica 7.1.2-6 から、Tuple Mover アルゴリズムの効率を向上させるために、MergeOutCache というジョブ分析ツールが追加されました。MergeOutCache を使用すると、Tuple Mover は、実行する必要のある mergeout ジョブをより迅速に判別できます。何百ものテーブルを持つ大きなカタログがある場合は、Vertica 7.1.2-6 以降の最新の Vertica パッチにアップグレードすることを検討してください。

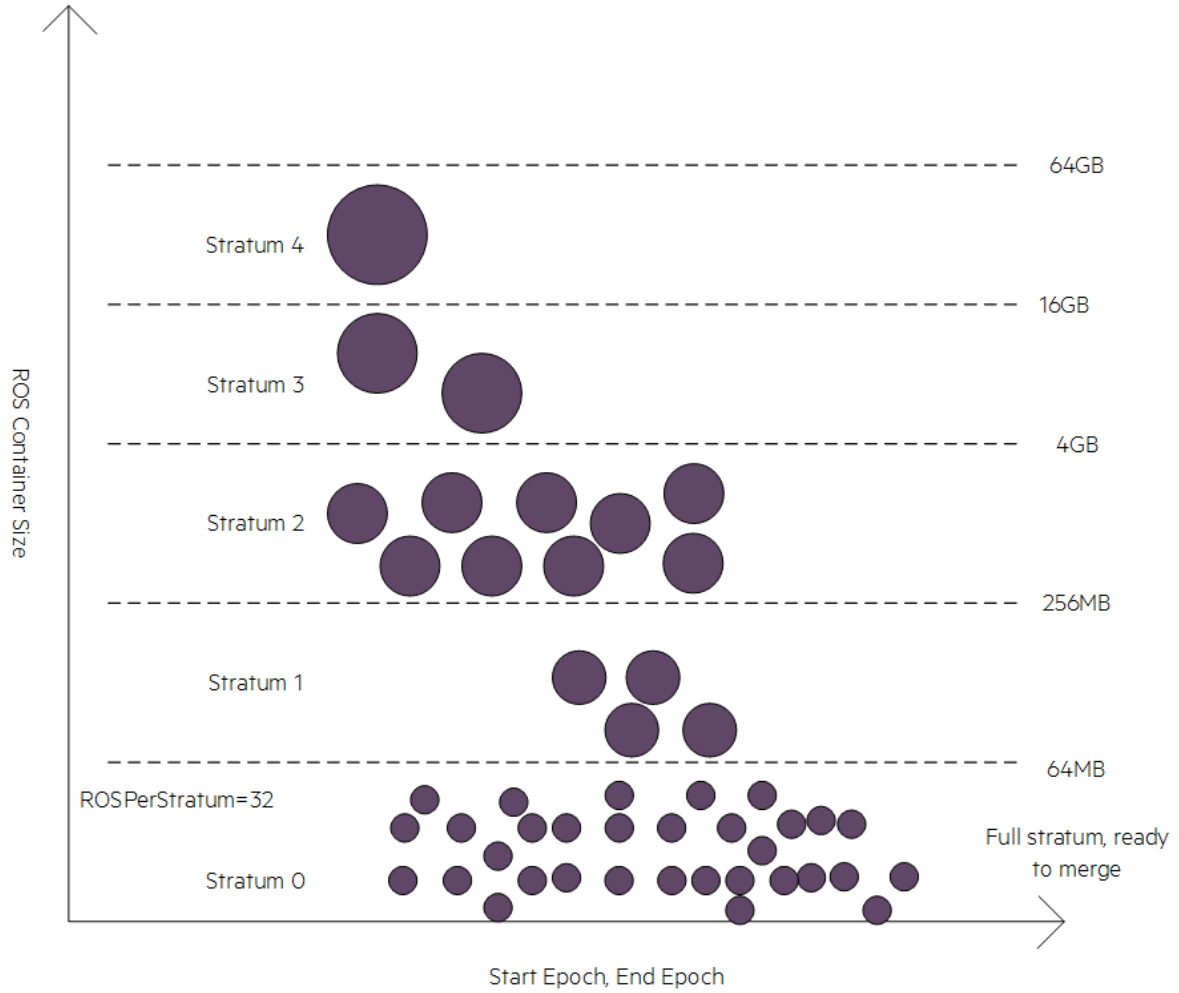
Mergeout に関するよくあるご質問

Tuple Mover mergeout の STRATA アルゴリズムはどのように動作するのでしょうか？

Tuple Mover の mergeout 処理は、STRATA ベースのアルゴリズムを使用します。このアルゴリズムは、データをロードするために使われたデータロードプロセスにかかわらず、mergeout を小さく何度も実施する対象となるかどうか検証するよう設計されています。このアルゴリズムによって、mergeout 処理では、パーティションを持たないテーブルおよびパーティション化されたテーブルのアクティブパーティションに対して、マージすべき ROS コンテナを選択します。

Vertica は、各アクティブなパーティションと、パーティション化されていないテーブルにひもづけられたプロジェクションの STRATA（階層）を定義しています。階層の数、各階層のサイズ、および階層内の ROS コンテナの最大数は、ディスクサイズ、メモリ、およびプロジェクションの列数に基づいて計算されます。

次の図は、アルゴリズムがサイズに基づいて ROS コンテナを階層に分類する仕組みを示しています。階層 0 内の ROS コンテナは、ROS サイズがごくわずかです（1 カラムあたり 1 MB 以下）。紫色の点は、ROS コンテナを表します。



大きな ROS コンテナをマージする前に小さな ROS コンテナをマージすることは、mergeout のアルゴリズムの最大の利点です。このアルゴリズムは、1 バイトからごくわずかな ROS チャンクサイズまでの階層 0 で始まり、上の階層に移動します。階層内の ROS コンテナの数が階層ごとに許可されている最大 ROS コンテナに等しいかそれ以上の値に達したかどうかをチェックします。（デフォルト値は 32 です）。アルゴリズムによって階層が一杯になったことがわかると、プロジェクションと階層は mergeout の対象としてマークされます。

Mergeout 処理では、階層内の ROS コンテナをすべて結合し、通常は次の階層に割り当てられる新しい ROS コンテナを生成します。階層 0 のものを除いて、mergeout 処理では、ROSPerStratum パラメーターの値に等しい ROS コンテナのみをマージします。階層 0 の場合、階層内のすべての ROS コンテナを 1 つの ROS コンテナにマージします。

デフォルトでは、mergeout 処理は 2 スレッドで動作します。通常、上位階層の大きな ROS コンテナの mergeout は、下位階層の ROS コンテナの mergeout よりも処理時間が長くなります。Mergeout スレッド 0 のみが、より高い階層および非アクティブパーティションで動作することができます。この制約により、mergeout スレッド 0 は、より高い階層で mergeout を実行するためにより多くの時間がかかるため、低い階層の ROS コンテナの蓄積を防止します。Mergeout スレッド 1 は、低いレイヤーの階層でのみ動作します。

注意：Mergeout スレッドをさらに追加すると、これらの追加スレッドは低い階層でのみ動作します。

非アクティブパーティション上で mergeout のスレッドはいくつ動作するのでしょうか？

Mergeout スレッド 0 は、非アクティブパーティションでのみ動作し、非アクティブパーティションの ROS コンテナを単一の ROS コンテナにマージします。スレッド 0 は、他のプロジェクションが STRATA アルゴリズムに基づいて mergeout 対象となっていない場合にのみ、非アクティブなパーティションで動作します。

Mergeout 処理は論理削除済みデータの物理削除も実行するのでしょうか？

Tuple Mover は、STRATA アルゴリズムに基づいて mergeout 対象となった ROS コンテナ内の AHM 以前に削除されたデータのみを物理削除します。また、非アクティブパーティションで削除されたデータも物理削除します。ただし、物理削除の対象にするには、非アクティブパーティションを持つ ROS コンテナ内の削除されたデータの割合が、PurgeMergeoutPercent パラメーターの値を超えなければなりません。このパラメータのデフォルト値は 20 です。

以下の構成パラメーターはいつどのように変更できるのでしょうか？

パラメーター	説明
`ROSPerStratum`	階層内の ROS コンテナの数。階層がデフォルト値の 32 に達すると、プロジェクションは mergeout 実行対象になります。より多くのレコードが mergeout 処理されることになるため、この値を小さくすると I/O 処理が増加します。しかし、その結果 ROS コンテナの数は減少します。
`MaxDVROSPerContainer`	単一の ROS コンテナに関連するデリートベクターの数がデフォルト値の 10 に達すると、Tuple Mover mergeout 処理は、デリートベクターをマージします。

プロジェクションのアクティブパーティションをどのように確認できるのでしょうか？

次のコマンドを使用して、特定のプロジェクションのアクティブパーティションを見つけることができます。

```
=>SELECT DISTINCT partition_key FROM strata;
 WHERE projection_name <projection_name>
 AND schema_name <schema>
```

ノードごとプロジェクションごとの ROS コンテナの数をどのように確認できるのでしょうか？

次のコマンドを使用して、ノードごとプロジェクションごとのコンテナ数を見つけることができます。