リバランスの理解 パート 2: リバランスの最適化

For Vertica 7.1.2

February, 2018

原文は[こちら]

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 2 です。第 2 部では、リバランスの前、中、後に取る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

Understanding Rebalancing, Part 1: What Happens During Rebalancing では、リバランス実行中の処理内容について説明します。

リバランス処理の開始

リバランス処理では、マシンリソースを実行中のクエリと共有する必要があり、リソース/ロックへの排他的なアクセスが必要となります。考えうる同時実行性の問題を回避するには、負荷の軽い期間にリバランスをスケジュールします。

ロード中に発生しうる同時実行性の問題の例を次に示します。

- リバランスが COPY 処理に必要なロックを所有している場合、COPY 処理はタイムアウトします。
- COPY がロックを所有している場合、COPY が完了するまで、リバランスは一時停止します。

Vertica のドキュメントに記載されている 3 つの方法のいずれかを使用して、リバランスを開始することができます。

- Rebalancing Data Using the Administration Tools UI
- Rebalancing Data Using Management Console
- Rebalancing Data Using SQL Functions

リバランスのモニタリング

リバランス処理をモニタリングして、問題なく進捗していることを確認します。以下のセクションでは、リバランス処理のモニタリング方法について説明します。

リバランス実行中のテーブルをモニタリング

Vertica がリバランス実行中のテーブルをモニタリングするには、次のコマンドを実行します。

```
=> SELECT * FROM rebalance_table_status;
```

DML 処理または DDL 処理で特定のテーブルのリバランスが妨げられた場合、次のエラーメッセージが表示されます。

```
ERROR 3007: DDL statement interfered with this statement
```

Vertica システムテーブルのモニタリング

リバランス中に次のシステムテーブルを確認してください。

- REBALANCE_TABLE_STATUS—各データベーステーブルに対して、以下の内容が含まれます。
  - 分離されたデータの量
  - 分離する必要があるデータの量
- REBALANCE_PROJECTION_STATUS—各プロジェクションに対して、以下の内容が含まれます。
  - 分離されたデータの量
  - 分離する必要があるデータの量

クラスターのリバランスをモニタリングするためのクエリ

このセクションでは、クラスターのリバランス処理をモニタリングするのに役立つサンプルクエリを紹介します。

アクティブなリバランス処理のモニタリング

次のクエリは、現在実行中のリバランス処理に関する情報を提供します。

- リバランス実行中のノード
- リバランス処理のセッション ID
- リバランスの開始時間。これにより、リバランスがどのくらいの期間実行中であるかを知ることができます。
- 現在実行中のリバランスのタスク

```
=> SELECT node_name, session_id, session_start_timestamp, description
 FROM system_sessions
 WHERE session_type = 'REBALANCE_CLUSTER'
 AND is_active;
 node_name |    session_id    |   session_start_timestamp     |
description
-----------+------------------+-------------------------------+--------------
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
----------------------------
   node001 | user-9956:0x2fb6 | 2016-02-11 21:36:16.045711-05 | Txn:
a000000000b167 'RebalanceUnsegmentedTask'
   node001 | user-9956:0x2fb6 | 2016-02-11 21:36:16.045711-05 | Txn:
a000000000b168 'CREATE PROJECTION public.dim7_node006
/*+basename(dim7),createtype(P)*/ ( a, b ) AS SELECT dim7.a, dim7.b FROM
public.dim7 ORDER BY dim7.a, dim7.b UNSEGMENTED NODE node006; '
```

```
   node001 | user-9956:0x2fc0 |  2016-02-11 21:36:16.13772-05 | Txn:
a000000000b16b 'Refresh: Evaluating which projection to refresh'
   node001 | user-9956:0x2fc0 |  2016-02-11 21:36:16.13772-05 | Txn:
a000000000b16c 'Refresh: (Table: public.dim7) (Projection:
public.dim7_node006)'
   node001 | user-9956:0x2fc0 |  2016-02-11 21:36:16.13772-05 | Txn:
a000000000b16f 'Refresh: through recovery (Table: public.dim7) (Projection:
public.dim7_node006)'
   node001 | user-9956:0x2fe1 | 2016-02-11 21:36:16.391713-05 | Txn:
a000000000b17a 'RebalanceUnsegmentedTask'
   node001 | user-9956:0x2fe1 | 2016-02-11 21:36:16.391713-05 | Txn:
a000000000b17b 'CREATE PROJECTION public.dim7_node007
/*+basename(dim7),createtype(P)*/(a,b) AS SELECT dim7.a, dim7.b FROM
public.dim7 ORDER BY dim7.a, dim7.b UNSEGMENTED NODE node007; '
   node001 | user-9956:0x2fe1 | 2016-02-11 21:36:16.391713-05 | Txn:
a000000000b17d 'CREATE PROJECTION public.dim7_node007
/*+basename(dim7),createtype(P)*/(a,b) AS SELECT dim7.a, dim7.b FROM
public.dim7 ORDER BY dim7.a, dim7.b UNSEGMENTED NODE node007; '
   node001 | user-9956:0x2fe3 |  2016-02-11 21:36:16.41582-05 | Txn:
a000000000b17e 'RebalanceUnsegmentedTask'
   node001 | user-9956:0x2fc0 |  2016-02-11 21:36:16.13772-05 | Txn:
a000000000b17f 'Refresh: through recovery (Table: public.dim7) (Projection:
public.dim7_node006)'
   node001 | user-9956:0x2fe3 |  2016-02-11 21:36:16.41582-05 | Txn:
a000000000b180 'CREATE PROJECTION public.dim7_node008
/*+basename(dim7),createtype(P)*/(a,b) AS SELECT dim7.a,dim7.b FROM
public.dim7 ORDER BY dim7.a, dim7.b UNSEGMENTED NODE node008; '
   node001 | user-9956:0x2fed | 2016-02-11 21:36:16.483142-05 | Txn:
a000000000b182 'Refresh: Evaluating which projection to refresh'
   node001 | user-9956:0x2fed | 2016-02-11 21:36:16.483142-05 | Txn:
a000000000b183 'Refresh: (Table: public.dim7) (Projection:
public.dim7_node007)'
   node001 | user-9956:0x2fc0 |  2016-02-11 21:36:16.13772-05 | Txn:
a000000000b184 'Refresh: through recovery (Table: public.dim7) (Projection:
public.dim7_node006)'
   node001 | user-9956:0x2ff6 | 2016-02-11 21:36:16.548277-05 | Txn:
a000000000b186 'Refresh: Evaluating which projection to refresh'
   node001 | user-9956:0x2fc0 |  2016-02-11 21:36:16.13772-05 | Txn:
a000000000b18a 'getRowCountsForProj'
   node001 | user-9956:0x2fc0 |  2016-02-11 21:36:16.13772-05 | Txn:
a000000000b18b 'analyze_row_count'
   node001 | user-9956:0x2fc0 |  2016-02-11 21:36:16.13772-05 | Txn:
a000000000b18c 'Refresh: gather PJP statistics (Table: public.dim7)
(Projection: public.dim7_node006)'
```

```
   node001 | user-9956:0x416c | 2016-02-11 21:37:34.654137-05 | Txn:
a000000000bb02 'RebalanceElasticTask'
...
```

リバランス処理の全体的な進捗状況をモニタリング

次のクエリは、現在実行中のリバランス処理の進捗状況をモニタリングします。

- 現在のプロジェクションをリバランスするために使用される方法。考えうる値は次の通りです。
  - REFRESH
  - REPLICATE
  - ELASTIC_CLUSTER
- 該当のメソッドのステータス
- プロジェクション数

```
=> SELECT rebalance_method Rebalance_method, Status, COUNT(*) AS Count
 FROM ( SELECT rebalance_method,
        CASE
        WHEN (
        separated_percent = 100
    AND transferred_percent = 100
        ) THEN 'Completed'
        WHEN (
               separated_percent <> 0
               and separated_percent <> 100
              )
           OR (
              transferred_percent <> 0
              AND transferred_percent <> 100
              ) THEN 'In Progress'
              ELSE 'Queued'
              END AS  Status
        FROM rebalance_projection_status
        WHERE is_latest
      ) AS tab
 GROUP BY 1, 2
 ORDER BY 1, 2;
 Rebalance_method | Status      | Count
------------------+-------------+-------
 ELASTIC_CLUSTER  | Completed   | 8
 ELASTIC_CLUSTER  | In Progress | 2
```

```
 ELASTIC_CLUSTER   | Queued       | 2
 REPLICATE         | Completed    | 50
(4 rows)
```

分散化されていないプロジェクションのリバランス状況をモニタリング

次のクエリは、現在リフレッシュされている分散化されていないプロジェクションに関する情報を返します。

- 現在実行中のリバランス処理のセッション ID
- リフレッシュされている分散化されていないプロジェクションの名前
- リフレッシュ処理の現在の状態
- プロジェクションをリフレッシュするために使用される方法
- リフレッシュがどのくらい進んでいるか。考えうるフェーズは次のとおりです。
    - Current
    - Historical

```
=> SELECT session_id, projection_name, refresh_status, refresh_method,
refresh_phase
 FROM projection_refreshes
 WHERE refresh_method = 'rebalance'
 AND is_executing;
 session_id        | projection_name | refresh_status | refresh_method |
refresh_phase
------------------+-----------------+----------------+----------------+------
---------
 user-9956:0x8cf0 | dim2_node004    | refreshing     | rebalance      |
current
 user-9956:0x8d24 | dim2_node005    | refreshing     | rebalance      |
historical
 user-9956:0x8d25 | dim2_node006    | refreshing     | rebalance      |
historical
(3 rows)
```

分散化されたプロジェクションのリバランス状況をモニタリング

次のクエリは、現在リフレッシュされている分散化されたプロジェクションに関する情報を返します。

- プロジェクション名
- プロジェクションをリフレッシュするために使用される方法
- 分離された ROS コンテナのパーセンテージ
- 宛先ノードに転送された ROS コンテナのパーセンテージ

```
=> SELECT projection_name, rebalance_method, separated_percent,
transferred_percent
 FROM rebalance_projection_status
 WHERE rebalance_method = 'ELASTIC_CLUSTER'
 AND (
 separated_percent <> 0
 AND separated_percent <> 100
 )
 OR (
 transferred_percent <> 0
 AND transferred_percent <> 100
 )
 AND is_latest;
 projection_name | rebalance_method | separated_percent | transferred_percent
-----------------+------------------+-------------------+--------------------
-
        fact5_b1 |  ELASTIC_CLUSTER |            100.00 | 13.23
        fact4_b1 |  ELASTIC_CLUSTER |             78.77 | 0.00
        fact5_b0 |  ELASTIC_CLUSTER |            100.00 | 13.63
(3 rows)
```

分散化されたプロジェクションの分離フェーズのアクティブな Tuple Mover 処理のモニタリング

次のクエリは、各ノード上の ROS コンテナを分離するプロセスの数に関する情報を返します。

- プロジェクション名
- ノード名
- 特定のプロジェクションの ROS コンテナの分離を Tuple Mover が開始した時間

```
=> SELECT TM.projection_name, TM.node_name, TM.operation_start_timestamp
 FROM tuple_mover_operations TM
 JOIN system_sessions
 USING (session_id)
 WHERE system_sessions.is_active
 AND session_type = 'REBALANCE_CLUSTER'
 AND operation_status = 'Running';
 projection_name | node_name | operation_start_timestamp
-----------------+-----------+-------------------------------
        fact1_b0 |   node008 | 2016-02-11 22:00:45.589359-05
        fact1_b1 |   node008 | 2016-02-11 22:00:45.589632-05
        fact1_b0 |   node008 | 2016-02-11 22:00:45.605845-05
```

```
        fact3_b0 |   node007 | 2016-02-11 21:57:36.339779-05
        fact3_b0 |   node008 | 2016-02-11 21:57:36.339917-05
        fact3_b1 |   node008 | 2016-02-11 21:57:36.340097-05
        fact1_b0 |   node004 | 2016-02-11 22:00:45.588915-05
        fact1_b1 |   node004 | 2016-02-11 22:00:45.58907-05
        fact3_b0 |   node005 | 2016-02-11 21:57:36.340339-05
 (10 rows)
```

ROS コンテナをモニタリング

次のクエリは、リバランス中に作成および削除された ROS コンテナに関する情報を返します。

- 該当のプロジェクションの ROS コンテナに対して実行されたアクション。考えうる値は次の通りです。
    - Created
    - Deleted
- プロジェクションの名前
- ROS コンテナに格納されている行数
- ROS コンテナの数

```
=> SELECT CASE
     WHEN (is_destroyed ) THEN 'deleted'
     ELSE 'created'
     END AS container, projection_name, SUM(row_count) AS rows_processed,
COUNT(*) n_containers
  FROM vs_rebalance_separated_storage_containers
  GROUP BY  1, 2
  ORDER BY 1, 2;
 container | projection_name | rows_processed | n_containers
-----------+------ ----------+----------------+--------------
   created |        fact3_b0 |         465975 | 10
   created |        fact3_b1 |         465975 | 10
   deleted |        fact1_b0 |         196273 | 730
   deleted |        fact1_b1 |         195780 | 740
   deleted |        fact2_b0 |         990830 | 71
   deleted |        fact2_b1 |         988365 | 72
   deleted |        fact5_b1 |        6140310 | 72
...
(20000000 rows)
```

次のクエリは、分散化されたプロジェクションと分散化されていないプロジェクションの間で転送された ROS コンテナに関する詳細情報を返します。

- プロジェクション名
- ソースノード名
- ターゲットノード名
- 転送された行数
- 転送されたバイト数

```
=> SELECT projection_name,
          from_node_name,
          to_node_name,
          SUM (row_count) rows_transfered, sum(size_in_bytes)
bytes_transferred
 FROM vs_rebalance_transferred_storage_containers
 GROUP BY 1, 2, 3
 ORDER BY 1, 2, 3;
projection_name | from_node_name | to_node_name | rows_transfered |
bytes_transferred
----------------+----------------+--------------+-----------------+----------
---------
   dim6_node007 |                |      node007 |         3900000 |
1972485567
   dim6_node008 |                |      node008 |         4000000 |
2023062120
   dim7_node001 |                |      node001 |          200000 | 101153106
   dim7_node002 |                |      node002 |          200000 | 101153106
       fact1_b1 |        node003 |      node002 |          104457 | 265520203
       fact1_b1 |        node003 |      node004 |           20671 | 52543973
       fact1_b1 |        node003 |      node005 |           31230 | 79391579
```

リバランスが完了したプロジェクションのリバランス実行時間のモニタリング

次のクエリは、各プロジェクションのリバランスに要した時間に関する情報を提供します。

- リバランスが実行されたノード
- スキーマ名
- プロジェクション名
- リバランス処理の開始時間
- プロジェクションのリバランスに要した時間(秒)

```
=> SELECT dc_rebalanced_projections.node_name,
          projection_schema,
          projection_name,
```

```
          start_time,
          time -start_time duration
 FROM dc_rebalanced_projections
 ORDER BY 5 DESC;
 node_name | projection_schema | projection_name | start_time
| duration
-----------+-------------------+-----------------+---------------------------
----+-----------------
   node001 |            public |        fact3_b1 | 2016-02-11
21:57:36.335312-05 | 00:03:42.695234
   node001 |            public |        fact3_b0 | 2016-02-11
21:57:36.335312-05 | 00:03:42.69519
   node001 |            public |        fact1_b1 | 2016-02-11
21:26:05.744551-05 | 00:03:38.692043
   node001 |            public |        fact1_b0 | 2016-02-11
21:26:05.744551-05 | 00:03:38.691965
   node001 |            public |        fact1_b1 | 2016-02-11
22:02:13.277392-05 | 00:03:35.077985
   node001 |            public |        fact1_b0 | 2016-02-11
22:02:13.277392-05 | 00:03:35.077949
```

リバランスとロックの競合

ETL ジョブの実行中にクラスターをリバランスすると、ロックの競合が発生する可能性があります。この競合により、ジョブまたはリバランス処理が失敗する可能性があります。

このトピックでは、Vertica が ETL ジョブを処理している間にリバランス処理を実行する場合に発生しうる競合の問題について説明します。また、それらを防ぐ方法についても説明します。

注意

一度に実行できるクラスター全体のリバランス処理は 1 つのみです。

競合の最小化

リバランス実行中に Tuple Mover とのロック競合を引き起こしうるデータベース処理は次の通りです。

- DELETE
- UPDATE
- DROP_PARTION
- SWAP_PARTITION_BETWEEN_TABLES
- MOVE_PARTITION_TO_TABLE

以下のセクションでは、競合を最小限に抑えるために、リバランスを診断、スケジュール、および管理する 3 つの方法について説明します。

- タイムアウト値を大きくする
- リバランスと Tuple Mover の処理を優先させる
- 手動でテーブルをリバランスする

タイムアウト値を大きくする

リバランスが ETL ジョブと競合することが考えられる場合は、LockTimeout 値を大きくすることができます。LockTimeout 構成パラメーターは、ETL ジョブがロックの解放を待つ時間を指定します。待機時間が LockTimeout 値を超えると、ETL ジョブはエラーを返します。デフォルト値は 300 秒(5 分)です。

LockTimeout 値を増やすと、リバランスが使用しているロックを取得しようとしている間にジョブがタイムアウトしない可能性が高まります。

次のクエリは、ETL ジョブがロックを保持していた時間が 5 分以上であるもの示しています。

```
=> SELECT DATE_TRUNC ('hour', grant_time), node_name,
COUNT(*) number_of_tx, MAX(time - grant_time) max_time_lock_held
FROM dc_lock_releases
WHERE time - grant_time>'5 min'
AND mode IN ('X', 'S', 'O')
AND object_name NOT LIKE 'ElasticCluster'
GROUP BY 1, 2
ORDER BY 4 desc;
date_trunc             | node_name | number_of_tx | max_time_lock_held
-----------------------+-----------+--------------+--------------------
2016-03-02 11:00:00-05 | node001   | 2            | 00:43:47.823902
2016-03-02 11:00:00-05 | node002   | 2            | 00:43:47.691664
2016-03-02 11:00:00-05 | node003   | 2            | 00:43:47.691906
2016-03-02 11:00:00-05 | node004   | 1            | 00:30:03.825279
2016-03-02 12:00:00-05 | node001   | 1            | 00:15:29.677898
2016-03-02 12:00:00-05 | node002   | 1            | 00:15:29.677404
2016-03-02 12:00:00-05 | node003   | 1            | 00:15:29.677366
2016-03-02 12:00:00-05 | node004   | 1            | 00:15:29.677111
(8 rows)
```

次のクエリは、特定のトランザクションが 5 分間以上ロックを保持していることを返します。

```
=> \x
Expanded display is on.
=> SELECT DISTINCT query_requests.transaction_id, statement_id, request
FROM dc_lock_releases JOIN query_requests USING (session_id)
WHERE time - grant_time > '5 min'
AND mode IN ('X','S')
AND object_name NOT LIKE 'ElasticCluster'
ORDER BY statement_id;
```

```
-[ RECORD 1 ]--+----------------------------------------
transaction_id | 45035996273756069
statement_id   | 1
request        | UPDATE /*+label(UPDATE_u1), DIRECT*/ u1 SET c200 = c200 - 1
where C200 < 100;
-[ RECORD 2 ]--+----------------------------------------
transaction_id | 45035996273756069
statement_id   | 2
request        | commit;
(2 rows)
```

これらのトランザクションとの競合を回避するには、次のいずれかを実行します。

- ETL ジョブと競合しない時間帯にリバランスを再スケジュールします。
- クラスターのリバランス中に ETL ジョブがタイムアウトしないように LockTimeout パラメーターを増やします。

```
=> SELECT GET_CONFIG_PARAMETER('locktimeout');
GET_CONFIG_PARAMETER
----------------------
 300
(1 row)
=> SELECT SET_CONFIG_PARAMETER('LockTimeOut, 600)
SET_CONFIG_PARAMETER
----------------------------
Parameter set successfully
(1 row)
```

リバランスの完了後、LockTimeout パラメーターを以前の値にリセットしてください。

リバランスと Tuple Mover の処理を優先させる

DML ジョブが、リバランスにロックされているテーブルにアクセスしようとしていたとします。デフォルトでは、DML ジョブはロックを取り、リバランスをキャンセルします。リバランスは、5 分後にテーブルへのアクセスを試み、リバランスを完了します。

リバランスを途切れることなく実行するには、DMLCancelTM 構成パラメーターを false に設定して、リバランス処理を優先します。この設定では、DML ジョブは進行中のリバランスが保持するロックを取ることができません。

DMLCancelTM を設定してリバランスを開始するには、次のように実施します。

```
=> SELECT SET_CONFiG_PARAMETER('DMLCancelTM', false);
....
=> SELECT REBALANCE_CLUSTER();
....
```

DML ジョブが重要な場合は、リバランスのために DMLCancelTM の値を false に変更しないでください。DMLCancelTM を true に設定すると、DML ジョブを実行できます。

重要な DML ジョブと競合しない時間帯に、リバランスを実行することを検討してください。リバランスが完了したら、必ず DMLCancelTM を true に戻してください。

```
=> SELECT SET_CONFiG_PARAMETER('DMLCancelTM', true);
```

手動でテーブルをリバランスする

多数のテーブルがあり、クラスターをリバランスするために数夜または週末が必要な場合、毎晩または週末に一定数のテーブルを手動でリバランスすることができます。

一度に 1 つ以上のテーブルを手動でリバランスすることができます。手動のリバランス中に競合が発生しないようにするには、ETL ジョブが実行されていないことを確認してください。

テーブルをリバランスするには、REBALANCE_TABLE 関数を呼び出します。

```
=> SELECT REBALANCE_TABLE('t0');
```

どのテーブルがリバランスされているか、リバランス実行中であるか、またはリバランスされていないかを調べるには、次のクエリを実行します。

```
=> SELECT table_name,
   CASE
   WHEN separated_percent + transferred_percent = 200 THEN 'REBALANCED'
   WHEN (separated_percent + transferred_percent) < 200
   AND (separated_percent + transferred_percent) > 0
   THEN 'REBALANCING' ELSE 'NOT REBALANCED YET'
   END status
   FROM rebalance_table_status WHERE is_latest;
 table_name | case
------------+----------------
         t0 | REBALANCED
         t1 | NOT REBALANCED YET
         t2 | REBALANCING
(3 rows)
```

リバランス実行中の競合エラー

次のセクションで説明するように、リバランス中に発生する可能性のある競合エラーがいくつかあります。

- `DDL statement interfered with this statement. Unavailable: lock table for query - Locking failure`
- `Staging table and target table do not match: Projections definition mismatch`
- `Unavailable: [Txn 0xa0000000010113] S lock table - timeout error Timed out`

スキーマの変更

次のような処理は、リバランス中に別のジョブによって発生する可能性があります。

- 列の追加または削除
- プロジェクションを追加または削除
- パーティションのスワップまたは移動

リバランスすると、ロックエラーが発生することがあります。このエラーは、別のジョブがリバランス処理によってすでにロックされているテーブルをロックしようとした場合に発生します。

```
ERROR 3007: DDL statement interfered with this statement
ERROR 5157: Unavailable: lock table for query - Locking failure: Timed out X
locking
Table:public.t0. T held by [user condor (RebalanceElasticTask)]. Your current
transaction isolation
level is SERIALIZABLE
```

パーティションのスワップ

リバランスされたテーブルとリバランスされていないテーブルの間でパーティションをスワップしようとすると、次のエラーが表示されます。

```
=> SELECT SWAP_PARTITIONS_BETWEEN_TABLES('t0', 1, 1, 't1');
ERROR 7121: Staging table and target table do not match: Projections
definition mismatch
```

どちらもリバランスされているか、あるいは、どちらもリバランスされていない 2 つのテーブル間でのみ、パーティションをスワップできます。

競合のエラーに関する情報の取得

エラーまたはロック競合のためにリバランスが失敗した場合、エラーに関する情報を取得するには、次のクエリを実行します。

```
=> SELECT time, session_id, error_level, node_name, log_message
   FROM dc_errors WHERE session_id IN
      (SELECT DISTINCT session_id
       FROM dc_session_starts
       WHERE session_type = 'REBALANCE_CLUSTER'
       )
   ORDER BY time DESC;
-[RECORD 1]-------------------------------
time        | 2016-03-05 10:47:08.517557-05
session_id  | eng-g9-046.verticac-2421955:0xad43
error_level | 20
node_name   | node001
```

```
log_message | Unavailable: [Txn 0xa0000000010113] S lock table - timeout
error Timed out S locking
Table:public.t1. I held
```

エラー後のリバランスの再開

エラーによりリバランスが失敗した場合、または DML 処理によってキャンセルされた場合、Vertica は 300 秒(5 分)後にリバランスを再試行します。これは、Vertica がリバランスを再開しようとするまでに 5 分待つことを意味します。

```
=> SELECT LIST_SERVICES('TM');
                    list_services
-------------------------------------------------------------------------------
Service: 'RebalanceCluster' is enabled , interval 300 second(s)
```

リバランス実行後

リバランス完了後、次の項目を確認します。

- リバランスが正常に完了しましたか?
- K-safety は正しいですか?
- ベースラインのパフォーマンスはどうですか?

成功または失敗

データベースのリバランス中に障害が発生した場合、再度リバランスを実行することができます。障害の原因が解決された場合、障害が発生した状態からリバランス処理が続行されます。ただし、データのリバランスが失敗すると、Vertica が自動的に削除することができない out-of-date のプロジェクションが発生する可能性があります。

このようなプロジェクションを見つけるには、次のように V_CATALOG.PROJECTIONS システムテーブルにクエリを実行します。

```
=> SELECT projection_name, anchor_table_name, is_prejoin, is_up_to_date
   FROM projections WHERE is_up_to_date = FALSE;
```

out-of-date のプロジェクションを削除するには、DROP PROJECTION を実行します。

Vertica 検証ユーティリティによるシステムパフォーマンスのモニタリング

Vertica には、システムのパフォーマンスをモニタリングするための 2 つのツールがあります。ただし、システムのパフォーマンスに大きな影響を与える可能性があるため、リバランシング実行時には実行しないでください。

- vioperf: ハードディスク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

ノードを追加した後で、これらのツールを使用して、新しく拡張されたクラスターのパフォーマンスベースラインを作成します。

詳細情報

Understanding Rebalancing, Part 1: What Happens During 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。