



Informatica® Intelligent Cloud Services  
July 2025

# データ統合のパフォーマンスのチューニング

© 著作権 Informatica LLC 2023, 2025

本ソフトウェアおよびマニュアルは、使用および開示の制限を定めた個別の使用許諾契約のもとでのみ提供されています。本マニュアルのいかなる部分も、いかなる手段（電子的複製、写真複製、録音など）によっても、Informatica LLC の事前の承諾なしに複製または転載することは禁じられています。

米政府の権利プログラム、ソフトウェア、データベース、および関連文書や技術データは、米国政府の顧客に配信され、「商用コンピュータソフトウェア」または「商業技術データ」は、該当する連邦政府の取得規制と代理店固有の補足規定に基づきます。このように、使用、複製、開示、変更、および適応は、適用される政府の契約に規定されている制限およびライセンス条項に従うものとし、政府契約の条項によって適当な範囲において、FAR 52.227-19、商用コンピュータソフトウェアライセンスの追加権利を規定します。

Informatica、Informatica Cloud、Informatica Intelligent Cloud Services、PowerCenter、PowerExchange、および Informatica ロゴは、米国およびその他の国における Informatica LLC の商標または登録商標です。Informatica の商標の最新リストは、Web (<https://www.informatica.com/trademarks.html>) にあります。その他の企業名および製品名は、それぞれの企業の商標または登録商標です。

本ソフトウェアまたはドキュメンテーション（あるいはその両方）の一部は、第三者が保有する著作権の対象となります。必要な第三者の通知は、製品に含まれています。

本マニュアルの情報は、予告なしに変更されることがあります。このドキュメントで問題が見つかった場合は、[infa\\_documentation@informatica.com](mailto:infa_documentation@informatica.com) までご報告ください。

Informatica 製品は、それらが提供される契約の条件に従って保証されます。Informatica は、商品性、特定目的への適合性、非侵害性の保証等を含めて、明示的または黙示的ないかなる種類の保証をせず、本マニュアルの情報を「現状のまま」提供するものとします。

発行日: 2025-12-01

# 目次

|  |    |
|--|----|
| <b>序文</b> .....  | 6  |
| Informatica のリソース.....                                   | 6  |
| Informatica マニュアル.....                                   | 6  |
| Informatica Intelligent Cloud Services Web サイト.....      | 6  |
| Informatica Intelligent Cloud Services コミュニティ.....       | 6  |
| Informatica Intelligent Cloud Services マーケットプレイス.....    | 7  |
| データ統合のコネクタのドキュメント.....                                   | 7  |
| Informatica ナレッジベース.....                                 | 7  |
| Informatica Intelligent Cloud Services Trust Center..... | 7  |
| Informatica グローバルカスタマサポート.....                           | 7  |
| <br><b>第 1 章 : データ統合のパフォーマンスのチューニングの概要</b> .....         | 8  |
| ベストプラクティス.....   | 8  |
| Secure Agent とクラウドリージョン.....                             | 8  |
| Secure Agent マシン.....                                    | 9  |
| Secure Agent のアップグレード.....                               | 13 |
| SQL ELT の最適化.....  | 13 |
| クラウドコネクタのパフォーマンス.....                                    | 13 |
| マッピングの設計と環境.....   | 14 |
| ボトルネック.....  | 15 |
| スレッド統計.....  | 16 |
| <br><b>第 2 章 : ターゲットの最適化</b> .....                       | 17 |
| ターゲットのボトルネックの特定.....                                     | 17 |
| ターゲットのボトルネックの除去.....                                     | 17 |
| <br><b>第 3 章 : ソースの最適化</b> .....                         | 18 |
| ソースのボトルネックの特定.....                                       | 18 |
| フィルタトランスフォーメーションの使用.....                                 | 18 |
| 読み取りテストマッピングの使用.....                                     | 18 |
| クエリの使用.....  | 19 |
| ソースのボトルネックの除去.....                                       | 19 |
| <br><b>第 4 章 : マッピングの最適化</b> .....                       | 20 |
| マッピングのボトルネックの特定.....                                     | 20 |
| マッピングのボトルネックの除去.....                                     | 20 |
| 区切りフラットファイルソースの最適化.....                                  | 21 |
| データプレビューの最適化.....  | 21 |
| フィルタの最適化.....  | 21 |
| データ型変換の最適化.....  | 22 |

|                                      |           |
|--------------------------------------|-----------|
| 式の最適化. . . . .                       | 22        |
| 共通ロジックの排除. . . . .                   | 22        |
| 集計関数呼び出しの最小化. . . . .                | 22        |
| ローカル変数での共通の式の置き換え. . . . .           | 23        |
| 文字列演算の代わりに数値演算の選択. . . . .           | 23        |
| LOOKUP の代わりに DECODE の選択. . . . .     | 23        |
| 関数の代わりに演算子の使用. . . . .               | 23        |
| IIF 関数の最適化. . . . .                  | 23        |
| ウィンドウ関数の最適化. . . . .                 | 24        |
| 式の評価. . . . .                        | 24        |
| アグリゲータトランスフォーメーションの最適化. . . . .      | 24        |
| 単純なカラム別のグループ化. . . . .               | 24        |
| ソート済み入力の使用. . . . .                  | 25        |
| 集計前のデータのフィルタリング. . . . .             | 25        |
| 接続されたフィールドの制限. . . . .               | 25        |
| 階層プロセッサトランスフォーメーションの最適化. . . . .     | 25        |
| ジョイナトランスフォーメーションの最適化. . . . .        | 26        |
| ルックアップトランスフォーメーションの最適化. . . . .      | 27        |
| ルックアップテーブルのキャッシュ. . . . .            | 27        |
| ルックアップ条件の最適化. . . . .                | 29        |
| ルックアップ行のフィルタリング. . . . .             | 29        |
| ルックアップテーブルのインデックス処理. . . . .         | 29        |
| 複数のルックアップの最適化. . . . .               | 29        |
| 機械学習トランスフォーメーションの最適化. . . . .        | 30        |
| ノーマライザトランスフォーメーションの最適化. . . . .      | 30        |
| ルータートランスフォーメーションの最適化. . . . .        | 30        |
| シーケンスジェネレータトランスフォーメーションの最適化. . . . . | 30        |
| ソータートランスフォーメーションの最適化. . . . .        | 31        |
| メモリの割り当て. . . . .                    | 31        |
| 作業ディレクトリの指定. . . . .                 | 31        |
| <b>第 5 章 : マッピングタスクの最適化. . . . .</b> | <b>32</b> |
| マッピングタスクのボトルネックの特定. . . . .          | 32        |
| マッピングタスクのボトルネックの除去. . . . .          | 32        |
| バッファメモリ. . . . .                     | 32        |
| キャッシュ. . . . .                       | 33        |
| 詳細ログ. . . . .                        | 34        |
| 詳細モードのベストプラクティス. . . . .             | 34        |
| <b>第 6 章 : 詳細クラスタの最適化. . . . .</b>   | <b>35</b> |
| 詳細クラスタのコンポーネント. . . . .              | 35        |
| ベストプラクティス. . . . .                   | 37        |
| 詳細クラスタノードの最適化. . . . .               | 38        |

|   |           |
|---|-----------|
| マスターノードの最適化. . . . .                    | 38        |
| ワーカーノードの最適化. . . . .                    | 38        |
| インスタンスタイプの最適化. . . . .                  | 38        |
| GPU インスタンスの使用. . . . .                  | 38        |
| Graviton インスタンスの使用. . . . .             | 39        |
| AMD チップセットの使用. . . . .                  | 39        |
| <b>第 7 章 : システムパフォーマンスの最適化. . . . .</b> | <b>40</b> |
| システムのボトルネックの特定. . . . .                 | 40        |
| Windows でのシステムのボトルネックの特定. . . . .       | 40        |
| Linux でのシステムのボトルネックの特定. . . . .         | 41        |
| システムのボトルネックの除去. . . . .                 | 41        |
| <b>索引. . . . .</b>                      | <b>42</b> |

# 序文

『データ統合のパフォーマンスチューニング』では、データ統合のマッピングのパフォーマンスを最適化する方法について説明します。

## Informatica のリソース

Informatica は、Informatica Network やその他のオンラインポータルを通じてさまざまな製品リソースを提供しています。リソースを使用して Informatica 製品とソリューションを最大限に活用し、その他の Informatica ユーザーや各分野の専門家から知見を得ることができます。

### Informatica マニュアル

Informatica マニュアルポータルでは、最新および最近の製品リリースに関するドキュメントの膨大なライブラリを参照できます。マニュアルポータルを利用するには、<https://docs.informatica.com> にアクセスしてください。

製品マニュアルに関する質問、コメント、ご意見については、Informatica マニュアルチーム ([infa\\_documentation@informatica.com](mailto:infa_documentation@informatica.com)) までご連絡ください。

### Informatica Intelligent Cloud Services Web サイト

Informatica Intelligent Cloud Services Web サイト (<http://www.informatica.com/cloud>) にアクセスできます。このサイトには、Informatica Cloud 統合サービスに関する情報が含まれます。

### Informatica Intelligent Cloud Services コミュニティ

Informatica Intelligent Cloud Services コミュニティを使用して、技術的な問題について議論し、解決します。また、技術的なヒント、マニュアルの更新情報、FAQ（よくある質問）への答えを得ることもできます。

次の Informatica Intelligent Cloud Services コミュニティにアクセスします。

<https://network.informatica.com/community/informatica-network/products/cloud-integration>

開発者は、次の Cloud 開発者コミュニティで詳細情報を確認したり、ヒントを共有したりできます。

<https://network.informatica.com/community/informatica-network/products/cloud-integration/cloud-developers>

## Informatica Intelligent Cloud Services マーケットプレイス

Informatica マーケットプレイスにアクセスすると、データ統合コネクタ、テンプレート、およびマップレットを試用したり購入したりできます。

<https://marketplace.informatica.com/>

## データ統合のコネクタのドキュメント

データ統合のコネクタのドキュメントには、マニュアルポータルからアクセスできます。マニュアルポータルを利用するには、<https://docs.informatica.com> にアクセスしてください。

## Informatica ナレッジベース

Informatica ナレッジベースを使用して、ハウツー記事、ベストプラクティス、よくある質問に対する回答など、製品リソースを見つけることができます。

ナレッジベースを検索するには、<https://search.informatica.com> にアクセスしてください。ナレッジベースに関する質問、コメント、ご意見の連絡先は、Informatica ナレッジベースチーム ([KB\\_Feedback@informatica.com](mailto:KB_Feedback@informatica.com)) です。

## Informatica Intelligent Cloud Services Trust Center

Informatica Intelligent Cloud Services Trust Center は、Informatica のセキュリティポリシーおよびリアルタイムでのシステムの可用性について情報を提供します。

Trust Center (<https://www.informatica.com/trust-center.html>) にアクセスします。

Informatica Intelligent Cloud Services Trust Center にサブスクライブして、アップグレード、メンテナンス、およびインシデントの通知を受信します。[Informatica Intelligent Cloud Services Status](#) ページには、すべての Informatica Cloud 製品の実稼働ステータスが表示されます。メンテナンスの更新はすべてこのページに送信され、停止中は最新の情報が表示されます。更新と停止の通知がされるようにするには、Informatica Intelligent Cloud Services の 1 つのコンポーネントまたはすべてのコンポーネントについて更新の受信をサブスクライブします。すべてのコンポーネントにサブスクライブするのが、更新を逃さないようにするための最良の方法です。

サブスクライブするには、[Informatica Intelligent Cloud Services Status](#) ページで **【サブスクライブして更新】** をクリックします。電子メール、SMS テキストメッセージ、Webhook、RSS フィード、またはこの 4 つの任意に組み合わせとして送信される通知を受信するという選択ができます。

## Informatica グローバルカスタマサポート

グローバルサポートセンターには、Informatica Network または電話でお問い合わせください。

Informatica Network でオンラインサポートリソースを検索するには、Informatica Intelligent Cloud Services のヘルプメニューで **【サポートにお問い合わせください】** をクリックして、**Cloud Support** ページに移動します。**Cloud Support** ページには、システムステータス情報とコミュニティディスカッションが記載されています。追加のリソースを検索する場合や電子メールで Informatica グローバルカスタマサポートに問い合わせる場合は、Informatica Network にログインし、**【サポートが必要な場合】** をクリックしてください。

Informatica グローバルカスタマサポートの電話番号は、Informatica の Web サイト <https://www.informatica.com/services-and-training/support-services/contact-us.html> に掲載されています。

## 第 1 章

# データ統合のパフォーマンスのチューニングの概要

パフォーマンスのチューニングは、パフォーマンスのボトルネックを除去することでマッピングのパフォーマンスを最適化することを目的として行います。

マッピングのパフォーマンスをチューニングするには、最初にパフォーマンスのボトルネックを特定して除去し、適切なパフォーマンスが得られるまでボトルネックの特定と除去を繰り返します。

パフォーマンスを向上させるための最善の方法を判断するには、反復的なプロセスが必要になります。一度に 1 つの変数を変更し、変更の前後でマッピングの時間を測定します。マッピングのパフォーマンスが向上しない場合は、元の設定に戻すことをお勧めします。

パフォーマンスの問題は、次のいずれかの理由で発生する可能性があります。

- ベストプラクティスに従っていない
- ターゲットのボトルネック
- ソースのボトルネック
- トランスフォーメーションのボトルネック
- システムのボトルネック

## ベストプラクティス

ベストプラクティスに従ってパフォーマンスを最適化します。

次のような領域に対して、ベストプラクティスの適用を検討してください。

- Secure Agent の環境と設定
- SQL ELT の最適化の使用
- クラウドコネクタのパフォーマンス
- マッピングの設計と環境

## Secure Agent とクラウドリージョン

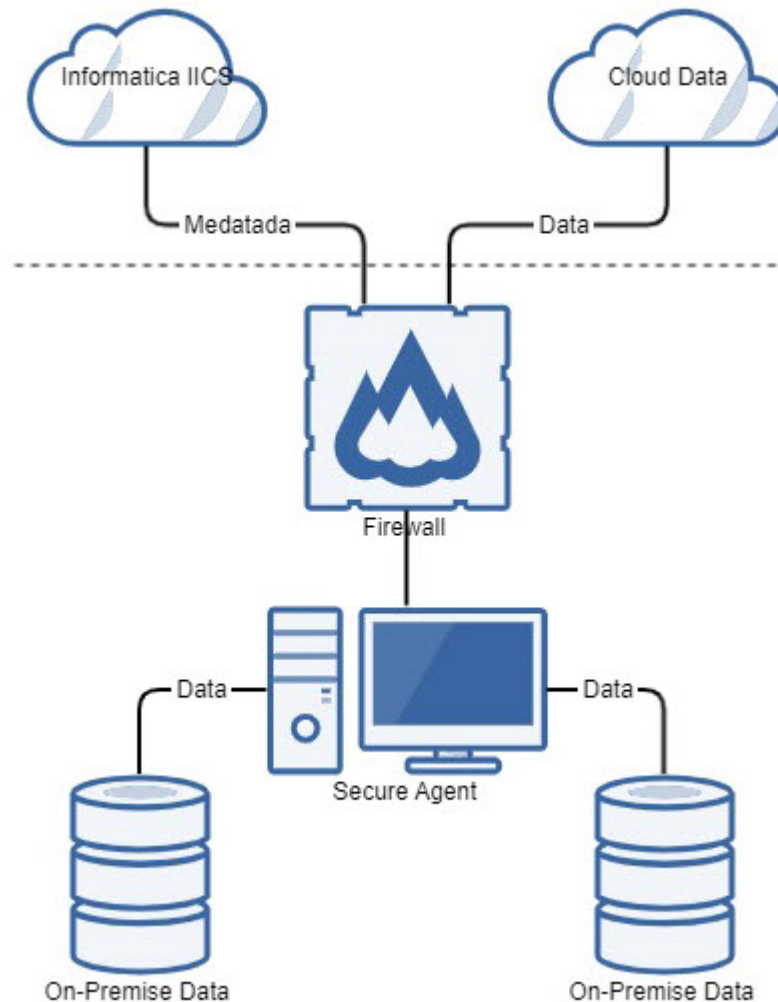
ネットワーク遅延によるパフォーマンスへの影響を回避するには、Secure Agent と、Secure Agent がデータのやり取りを行うコンポーネントを同じリージョン内に配置します。例えば、クラウドデータウェアハウスの



エンドポイントが AWS 米国西部リージョンにある場合は、米国西部リージョン内にある Secure Agent マシンを検索します。

Secure Agent とコンポーネントがオンプレミスコンポーネントを使用せずに同じクラウド環境にデプロイされている場合は、それらを同じ VPC/サブネット上に配置します。また、可用性ゾーンが同じリージョン内に存在している必要があります。

次の図に、Informatica Intelligent Cloud Services Secure Agent と、エージェントがデータのやり取りを行うコンポーネントを示します。



## Secure Agent マシン

クラウドデータレイクからクラウドデータウェアハウスへのマッピングには、ディスク入出力（I/O）とネットワーク帯域幅で高スループットをサポートするハードウェアを選択します。

Secure Agent がインストールされているマシンが次の要件を満たしていることを確認します。

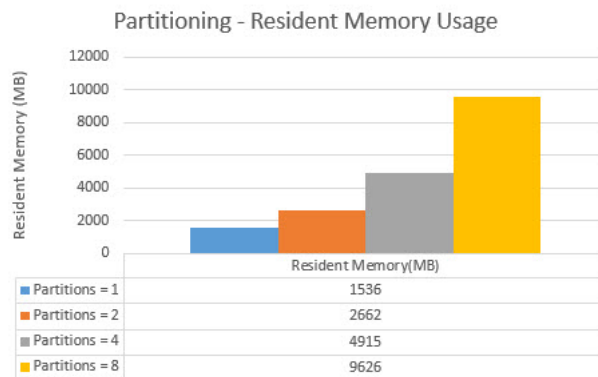
- 500 MBps の最小ディスクスループットをサポート
- 2.5 Gbps の最小ネットワークスループットをサポート

## Secure Agent マシンのサイズ変更要件

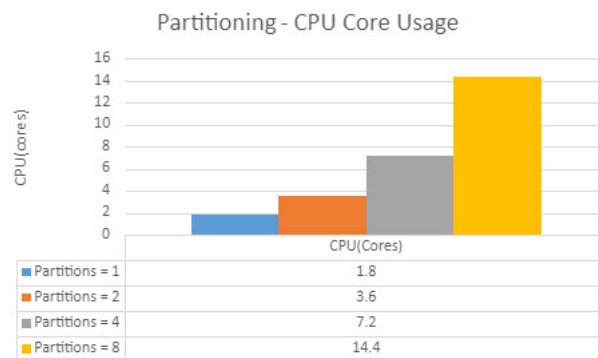
最適なパフォーマンスを得るには、Secure Agent マシンのメモリ要件を考慮します。

一般的なクラウドデータレイクからクラウドデータウェアハウスへのマッピングには、約 7.5 GB のデータサイズに対して最大 3 個の CPU コアと 1 GB の JVM ヒープメモリが必要となる場合があります。JVM ヒープメモリのデフォルト値は 64 MB です。DTM バッファブロックサイズとバッファプールサイズを追加すると、メモリフットプリントが増加します。

次のグラフに、フラットファイルからクラウドデータウェアハウスへのパススルーマッピングの物理メモリ（常駐メモリ）使用量を、パーティションの数に関連させて示します。デフォルトのバッファブロックサイズは 100 MB に設定されており、JVM ヒープメモリは 1 GB に設定されています。



クラウドデータレイクとクラウドデータウェアハウスのマッピングにパーティションを追加すると、CPU コアの要件が直線的に増加します。次のグラフに、パーティション数が増加した場合の、フラットファイルからクラウドデータウェアハウスへのパススルーマッピングのコアの CPU 消費率を示します。



パフォーマンスを向上させるには、maxDTMProcesses カスタムプロパティと JVM オプションを設定します。

## 詳細モードの Secure Agent マシンのサイズ変更要件

詳細モードでマッピングの処理を行う Secure Agent の場合は、最適なパフォーマンスを実現するための Secure Agent マシンの要件を考慮してください。

Secure Agent には少なくとも 4 個の CPU、16 GB のメモリ、および 100 GB のディスク容量が必要です。最適な処理を行うには、SSD を使用してください。

次の表に、Secure Agent が処理する同時 Spark タスクの数に基づいた最適な Secure Agent の構成を示します。

| 同時 Spark タスクの数       | CPU とメモリ  | JVM ヒープサイズ                 |
|----------------------|---|----------------------------|
| 0-250 件の Spark タスク   | 8 個の CPU と 32 GB のメモリ   | 2GB                        |
| 250-500 件の Spark タスク | 16 個の CPU と 64 GB のメモリ<br>AWS では、8 個の CPU と 32 GB のメモリを使用します。 | 4GB<br>AWS では 2 GB を使用します。 |

JVM ヒープサイズはデフォルトで 2 GB に設定されています。メモリ不足エラーを回避するには、ヒープサイズを増やします。

## maxDTMProcesses プロパティ

maxDTMProcesses カスタムプロパティを設定して、パフォーマンスを向上させます。

デフォルトでは、Secure Agent は 2 つのマッピングタスクの実行をスケジューリングできます。タスクが 3 つ以上ある場合は、追加のタスクがキューに格納され、スロットが使用可能になったときに実行するようにスケジューリングされます。これにより、Secure Agent マシンの容量が十分に活用されなくなる可能性があります。

Secure Agent マシンの CPU 容量をより有効に活用し、より高い同時実行性を実現するには、データ統合サーバーの maxDTMProcesses カスタムプロパティを並列タスクの数に設定します。例えば、このプロパティを 16 に設定すると、16 個のタスクを同時に実行できます。

次の図に、Administrator のエージェントの詳細ページにある maxDTMProcesses カスタムプロパティの設定を示します。

| Custom Configuration Details |        |          |                 |       |                          |
|------------------------------|--------|----------|-----------------|-------|--------------------------|
| Service                      | Type   | Sub-type | Name            | Value | Sensitive                |
| Data Integration Server      | Tomcat |          | maxDTMProcesses | 16    | <input type="checkbox"/> |

推奨される maxDTMProcesses 値は、エージェントが実行するジョブの接続タイプによって異なります。

- ジョブでファイルベース接続または ODBC 接続が使用されている場合は、論理 CPU の数に 0.75 をかけた値を使用します。  
例えば、Secure Agent マシン上に 24 個の論理 CPU がある場合は、この値を 18 に設定します。
- ジョブでクラウドデータレイクまたはクラウドデータウェアハウスコネクタが使用されている場合は、論理 CPU の数に 0.33 をかけた値を使用します。  
例えば、Secure Agent マシン上に 24 個の論理 CPU がある場合は、この値を 8 に設定します。

maxDTMProcesses カスタムプロパティを設定するときは、次のガイドラインに注意してください。

- このプロパティを高い値に設定すると、タスク実行の並列処理が向上しますが、タスク実行時間のパフォーマンスのボトルネックが発生したり、マッピングタスクのジョブディスパッチ遅延が増加したりする可能性があります。
- プロパティをより低い値に設定すると、マッピングジョブは、スロットを取得するために前のジョブが完了するまで待機する可能性があります。
- maxDTMProcesses プロパティを設定する場合は、使用許諾契約の条件を超えないようにしてください。例えば、組織が 3 つの Secure Agent のライセンスを取得しており、各エージェントを最大 4 個の CPU で実行できる場合は、このプロパティを 12 より大きい値に設定しないでください。

これらのガイドラインを開始点として使用し、正しい値の設定を試行します。

Secure Agent のプロパティの設定に関する詳細については、「ランタイム環境」を参照してください。

## INFA\_MEMORY および JVM オプション

INFA\_MEMORY プロパティと JVM オプションを設定して、最適なパフォーマンスを実現し、Java ヒープおよびメモリ関連のエラーを回避できます。これらのプロパティは、Administrator のエージェントの詳細ページで定義します。

INFA\_MEMORY および JVM オプションは、次の形式の値を使用して設定できます。

| 形式                   | 説明  |
|----------------------|---|
| -Xms**m              | Java プロセスの開始時に JVM に割り当てられるメモリの初期量。<br>これは初期値であるため、Xms 値は 64m や 128m などの小さい値にすることができます。Java プロセスは、必要に応じてさらに多くのスペースを割り当てます。  |
| -Xmx****m            | JVM がヒープとして割り当てることができるメモリの最大量。Java プロセスが開始されると、このプロセスはオブジェクトを保存するためのスペースの割り当てを継続します。割り当ては最大設定に達するまで継続されます。<br>すべての Java オブジェクトとクラスを保持するために必要な値を設定します。64 ビットのエージェントでは、この値は約 1024M または 2048M になります。 |
| -XX:MaxPermSize=***m | JVM が一度に使用できる永続スペースの最大量。指定された量を超える量が JVM で必要となった場合、Java プロセスは失敗します。<br>この値は平均 512M に設定できます。ただし、この値は-Xmx 値よりも小さい値である必要があります。permGen スペースに関するエラーが表示された場合は、MaxPermSize 値を増やします。                      |

## INFA\_MEMORY

**[システム構成の詳細]** セクションのエージェント詳細ページで INFA\_MEMORY プロパティを定義します。データ統合サービスと Tomcat JRE タイプを選択し、INFA\_MEMORY プロパティを定義します。

次の図に、エージェントの詳細ページの INFA\_MEMORY プロパティを示します。

図 1.

| ▼ System Configuration Details |                           |                    |                          |
|--------------------------------|---------------------------|--------------------|--------------------------|
| Service:                       | Data Integration Server ▼ |                    |                          |
| Type:                          | Tomcat JRE ▼              |                    |                          |
| Type                           | Name                      | Value              | Sensitive                |
| Tomcat JRE                     | INFA_SSL                  |                    | <input type="checkbox"/> |
| Tomcat JRE                     | INFA_MEMORY               | "-Xms32m -Xmx512m" | <input type="checkbox"/> |

## JVM オプション

**[システム構成の詳細]** セクションのエージェントの詳細ページで JVM オプションを定義します。データ統合サービスと DTM タイプを選択し、JVMOption プロパティを定義します。

各 JVMOption プロパティには 1 つの JVM 属性を含めることができます。最大 5 つの JVMOption プロパティを定義できます。すでに 5 つの JVMOption プロパティがある場合は、カスタムプロパティとしてさらにプロパティを作成することもできます。

-Xmx プロパティにより、ヒープサイズを決定します。デフォルト値は 64MB です。複数のパーティションを含む大量のデータを処理するマッピングの場合、デフォルト値では Java ヒープ領域が十分ではないため、マッピングが失敗する可能性があります。このプロパティの推奨値は 2024MB です。

次の図に、エージェントの詳細ページの JVMOption1 プロパティの設定を示します。

| ▼ System Configuration Details |                           |                   |                          |
|--------------------------------|---------------------------|-------------------|--------------------------|
| Service:                       | Data Integration Server ▼ |                   |                          |
| Type:                          | DTM ▼                     |                   |                          |
| Type                           | Name                      | Value             | Sensitive                |
| DTM                            | JVMClassPath              | 'pmserversdk.jar' | <input type="checkbox"/> |
| DTM                            | JVMOption1                | '-Xmx2024m'       | <input type="checkbox"/> |

## Secure Agent のアップグレード

組織で複数のサービスとコネクタを使用している場合は、Secure Agent グループが使用される頻度が高くなる可能性があります。Secure Agent のアップグレードのパフォーマンスを向上させるために、サービスとコネクタの Secure Agent グループを無効にすることができます。

## SQL ELT の最適化

クラウドデータレイクとクラウドデータウェアハウスの統合の場合、データ統合は、データパイプライン処理にクラウドエコシステムまたはクラウドデータウェアハウス API を活用する機会をインテリジェントに特定します。

データ統合は、SQL ELT の最適化を使用して、処理をクラウドエコシステムまたはクラウドデータウェアハウスにプッシュダウンします。これにより、データがソースの近くの部分で処理されるため、データ処理のパフォーマンスが向上し、企業のデータ転送コストが軽減されます。

データ統合は、クラウドデータウェアハウス間の統合に対する完全な SQL ELT の最適化をサポートします。データ統合は、通常、追加の処理時間とデータ転送コストが発生するデータ転送を行わずに、クラウドデータウェアハウスが実行する同等の最適化された SQL クエリにマッピングロジックを変換します。データ統合では、追加のリソースを用いずにクラウドデータウェアハウスのコンピューティング能力を使用してデータが処理されるため、優れたコンピューティング効率が実現されます。

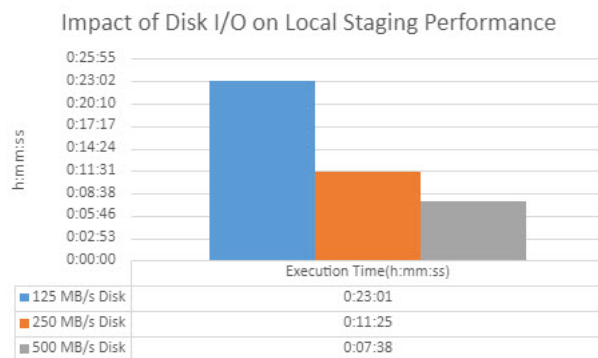
SQL ELT の最適化がサポートされている場合は、クラウドデータレイクからクラウドデータウェアハウスへの統合、またはクラウドデータウェアハウスからクラウドデータウェアハウスへの統合に対して SQL ELT の最適化を使用するようにしてください。

## クラウドコネクタのパフォーマンス

クラウドデータレイクとクラウドデータウェアハウスコネクタは、データのロードとアンロードのパフォーマンスを最適化するように設計されています。

これらのコネクタは、エンドポイントにアップロードする前、またはエンドポイントからデータをダウンロードした後に、Informatica がデータをディスク上にローカルにステージングするという共通した設計パターンを持ちます。このステージングプロセスはディスクを大量に使用する操作であるため、十分な CPU リソースとディスク I/O リソースを必要とします。SQL ELT の最適化が使用されていない場合は、クラウドデータレイクからクラウドデータレイクへ、クラウドデータレイクからクラウドデータウェアハウスへ、およびクラウドデータウェアハウスからクラウドデータウェアハウスへといったすべての統合において、この点に留意してください。

次のグラフに、同時クラウドデータウェアハウスマッピングのパフォーマンスと、持続的なディスク I/O がマッピングに与える影響を示します。



Informatica では、パーティション化が有効になっているか、同時実行が行われているデータ統合ワークロードに対して、500 Mbps のディスクスループットを備えたストレージデバイスを推奨しています。これらのエンドポイントに対するさまざまなチューニングオプションの詳細については、適切なコネクタガイドを参照してください。一部のコネクタについては、Informatica Knowledge Base でパフォーマンスチューニングの記事を確認することもできます。

## マッピングの設計と環境

マッピングを設計するときは、マッピングのパフォーマンスを最適化するためのベストプラクティスに従ってください。

次のベストプラクティスを考慮してください。

### データ量を減らす。

- 各カラムのフィールド精度を必要な長さまで減らします。
- カラムの数を減らして、接続されていないフィールドを削除します。
- 行数を減らして、ソースまたはフィルタトランスフォーメーションでソースフィルタを使用します。
- マッピングの初期の部分でフィルタトランスフォーメーションを使用して、不要なデータを削除します。

### ソースのパーティション化を有効にする。

可能な場合は常にソースのパーティション化を有効にします。マッピングタスクは、ソースデータをパーティションに分割し、それらのパーティションを同時に処理します。

### データ変換を最適化する。

- ソースからターゲットまでの一貫性を維持します。可能な限り、パイプライン全体で同じデータ型と精度を維持するようにしてください。
- 可変長データ型の場合は、できるだけ小さい精度を使用します。
- 操作が行われない場合は、日付の代わりに String データ型を使用します。
- 不要なデータ型変換を除去します。例えば、Integer カラムから Decimal カラムへデータを移動した後、再び Integer カラムにそのデータを戻すマッピングの場合は、不要なデータ型変換によってパフォーマンスが低下します。
- ルックアップトランスフォーメーションとフィルタトランスフォーメーションを使用して比較を実行する場合は、他のデータ型の代わりに可能な限り整数値を使用します。

### ローカルのフラットファイルステージングを使用する。

マッピングがクラウドデータウェアハウスに書き込みを行う場合、またはクラウドデータウェアハウスから読み取りを行う場合は、クラウドデータウェアハウスエンドポイントに書き込みを行う前あるいはクラウドデータウェアハウスから読み取りを行う前にデータを一時フォルダにローカルでステージングするように Secure Agent を設定することで、マッピングのパフォーマンスを最適化することができます。

Secure Agent のプロパティで、Tomcat のステージングプロパティ `INFA_DTM_STAGING_ENABLED_CONNECTORS` をクラウドデータウェアハウスコネクタのプラグイン ID に設定します。データ統合はフラットファイルをローカルで作成してデータをステージングし、ステージングファイルからデータウェアハウスターゲットにデータをロードするか、データウェアハウスソースからデータをアンロードしてローカルでステージングします。詳細については、個々のクラウドコネクタのパフォーマンスチューニングガイドを参照してください。

### ハードウェアをチューニングする。

例えば、ネットワーク速度を改善し、複数の CPU を使用します。

### Secure Agent 仮想マシンのインスタンスタイプを考慮する。

リソース要件に基づいて、Amazon Elastic Compute Cloud (EC2)、Azure Virtual Machine (Azure VM)、Google Cloud Platform (GCP) などのパフォーマンスに優れたクラウドインスタンスを選択します。

## ボトルネック

パフォーマンスのチューニングを行う場合は、最初の手順としてパフォーマンスのボトルネックを特定します。パフォーマンスのボトルネックは、ソースデータベースとターゲットデータベース、マッピング、マッピングタスク、およびシステムで発生する可能性があります。

ボトルネックのチューニングを行うことで、別のボトルネックが明らかになる場合もあります。パフォーマンスのボトルネックを特定してそのボトルネックを解消した後に、適切な水準のパフォーマンスが得られるまでボトルネックの特定と除去を続けるというのが基本的な方針となります。

パフォーマンスのボトルネックは、以下の順に調査します。

1. Target
2. Source
3. マッピング
4. マッピングタスク
5. System

次の方法でパフォーマンスのボトルネックを特定します。

### テストマッピングを実行する。

テストマッピングを設定して、フラットファイルソースから読み取りを行うかフラットファイルターゲットに書き込みを行うことで、ソースとターゲットのボトルネックを特定できます。

### スレッド統計を分析する。

スレッド統計を分析して、最適なパーティションの数を決定します。

### システムパフォーマンスを監視する。

システム監視ツールを使用して、CPU 使用率、入出力待ち、およびページングについての情報を表示することで、システムのボトルネックを特定できます。



## スレッド統計

セッションログのスレッド統計を使用すると、ソース、ターゲットまたはトランスフォーメーションのボトルネックを特定できます。

デフォルトでは、マッピングは1つの読み取りスレッド、1つのトランスフォーメーションスレッド、および1つの書き込みスレッドを使用します。ビジー率の最も高いスレッドが、マッピングの実行のボトルネックです。特定のトランスフォーメーションがボトルネックであると判断される場合は、そのトランスフォーメーションをチューニングすることができます。

セッションログには次のスレッド統計が表示されます。

### ランタイム

スレッドの実行時間。

### アイドル時間

スレッドがアプリケーション内で他のスレッドの処理を待機する時間を含む、スレッドがアイドル状態である時間。アイドル時間には、データ統合サーバーによってスレッドがブロックされた時間が含まれますが、オペレーティングシステムによってブロックされた時間は含まれません。

### ビジー時間

次の式に従って計算された、スレッドがビジー状態である実行時間の割合。

$$(\text{Run time} - \text{idle time}) / \text{run time} \times 100$$

合計実行時間が60秒以下などのように短い場合は、ビジー率が高くても無視することができます。これは必ずしもボトルネックを示しているわけではありません。

トランスフォーメーションスレッド内でボトルネックとなっているトランスフォーメーションを特定するには、スレッド作業時間分析で各トランスフォーメーションのビジー率を確認します。ソース、ターゲット、またはトランスフォーメーションがボトルネックとして表示されない場合は、タスクのプロパティがボトルネックの原因である可能性があります。



## 第 2 章

# ターゲットの最適化

ターゲットのボトルネックを除去することで、パフォーマンスを最適化できます。最も一般的なパフォーマンスのボトルネックは、マッピングがターゲットデータベースへの書き込みを行うときに発生します。

ネットワーク遅延、データベースまたはデータウェアハウスに使用されるターゲットインスタンスタイプのパフォーマンスの問題、または高負荷操作時の問題によって、ターゲットのボトルネックが発生する可能性があります。

## ターゲットのボトルネックの特定

フラットファイルターゲットを使用してマッピングのパフォーマンスをテストし、スレッド統計を使用してターゲットのボトルネックを特定します。

ターゲットのボトルネックを特定するには、次のタスクを実行します。

- フラットファイルターゲットに書き込むマッピングのコピーを設定します。このマッピングによってパフォーマンスが著しく向上する場合は、ターゲットにボトルネックがあります。マッピングですでにフラットファイルターゲットへの書き込みが行われている場合、ターゲットにはボトルネックがないと判断できます。
- セッションログのスレッド統計を確認します。マッピングタスクでトランスフォーメーションスレッドまたは読み取りスレッドよりも書き込みスレッドの処理に時間がかかっている場合は、ターゲットにボトルネックがあります。

## ターゲットのボトルネックの除去

ベストプラクティスと、ターゲットのクラウドデータベースまたはデータウェアハウスのパフォーマンスは、ターゲットのボトルネックの発生に影響を与える可能性があります。

ターゲットのボトルネックを除去するには、次のタスクを実行します。

- クラウドとマッピングの設計のベストプラクティスに従っていることを確認します。
- ターゲットのクラウドデータベースまたはデータウェアハウスが適切に実行されていることを確認します。パフォーマンスが低い場合は、マッピングのパフォーマンスが低下する可能性があります。

## 第 3 章

# ソースの最適化

ソースのボトルネックを除去することで、パフォーマンスを最適化できます。マッピングタスクがソースから読み取りを行うときに、パフォーマンスのボトルネックが発生する可能性があります。

非効率的なクエリ、ネットワーク遅延、データベースまたはデータウェアハウスソースに使用されるインスタンスタイプのパフォーマンスの問題により、ソースのボトルネックが発生する可能性があります。

## ソースのボトルネックの特定

セッションログのスレッド統計を確認して、ソースがボトルネックになっているかどうかを判断できます。マッピングでトランスフォーメーションスレッドまたは書き込みスレッドよりも読み取りスレッドの処理に時間がかかっている場合は、ソースにボトルネックがあります。

マッピングがクラウドデータベースまたはデータウェアハウスソースから読み取りを行う場合は、次の方法を使用してソースのボトルネックを特定します。

- Filter トランスフォーメーションを使用する。
- テストマッピングを使用する。
- データベースクエリーを使用する。

マッピングでフラットファイルソースからの読み取りが行われている場合、ソースにはボトルネックがないと判断できます。

## フィルタトランスフォーメーションの使用

マッピング内でフィルタトランスフォーメーションを使用して、ソースデータの読み取りにかかる時間を測定できます。

各ソースの後にフィルタトランスフォーメーションを追加します。フィルタ条件を `false` に設定すると、データがフィルタトランスフォーメーションによって処理されなくなります。新しいマッピングの実行時間が以前と同じ場合は、ソースにボトルネックがあります。

## 読み取りテストマッピングの使用

読み込みテストマッピングを作成して、ソースのボトルネックを特定できます。読み込みテストマッピングでは、マッピング内のトランスフォーメーションを削除することによって、読み込みクエリーを見つけ出します。

読み込みテストマッピングを作成するには、以下の手順を実行します。

1. 元のマッピングのコピーを作成します。

2. マッピングのコピーには、ソーストランスフォーメーションとカスタム結合またはクエリのみを保持するようにします。
3. 他のすべてのトランスフォーメーションは削除します。
4. ソースをフラットファイルターゲットに接続します。

読み取りテストマッピングを実行します。マッピングのパフォーマンスが元の実行のパフォーマンスと変わらない場合は、ソースにボトルネックがあります。

## クエリの使用

ボトルネックを特定するには、ソースデータベースまたはデータウェアハウスに対して読み取りクエリを直接実行します。

読み取りクエリをセッションログから直接コピーします。クエリツールを使用してソースに対してクエリを実行します。Windows では、クエリの結果をファイルにロードすることができます。Linux では、クエリの結果を `/dev/null` にロードすることができます。

クエリの実行時間およびクエリが最初の行を返すまでにかかる時間を測定します。

## ソースのボトルネックの除去

ベストプラクティスと、ソースクラウドデータベースまたはデータウェアハウスのパフォーマンスは、ソースでのボトルネックの発生に影響を与える可能性があります。

ソースのボトルネックを除去するには、次のタスクを実行します。

- クラウドとマッピングの設計のベストプラクティスに従っていることを確認します。
- ソースクラウドデータベースまたはデータウェアハウスが適切に実行されていることを確認します。パフォーマンスが低い場合は、マッピングのパフォーマンスが低下する可能性があります。管理者はクエリを最適化してパフォーマンスを最適化することができます。

## 第 4 章

# マッピングの最適化

マッピングの設計時にボトルネックを除去し、ベストプラクティスに従うことで、マッピングのパフォーマンスを最適化できます。

## マッピングのボトルネックの特定

ソースまたはターゲットにボトルネックがないことが確認できた場合は、マッピングにボトルネックが存在する可能性があります。スレッド統計とフィルタトランスフォーメーションを使用して、マッピングのボトルネックを特定します。

マッピングのボトルネックを特定するには、次のタスクを実行します。

- セッションログのスレッド統計を確認します。マッピングで書き込みスレッドまたは読み取りスレッドよりもトランスフォーメーションスレッドの処理に時間がかかっている場合は、トランスフォーメーションにボトルネックがあります。
- 各ターゲットトランスフォーメーションの前にフィルタトランスフォーメーションを追加します。データがターゲットにロードされないように、フィルタ条件を `false` に設定します。新しいマッピングタスクの実行時間が元のマッピングタスクの実行時間と同じ場合は、マッピングにボトルネックがあります。

## マッピングのボトルネックの除去

マッピングのボトルネックを除去するには、マッピングを最適化します。

マッピングレベルの最適化を実行するには時間がかかりますが、パフォーマンスを大幅に向上させることができます。マッピングレベルの最適化は、ターゲットおよびソースの最適化が完了した後に実行するようにします。

マッピングの最適化を行う場合、通常は、マッピング内のトランスフォーメーションの数を減らして、トランスフォーメーション間の不要なリンクを削除します。処理されるデータの容量を最大化するには、トランスフォーメーションと式の数できるだけ少なくした状態でマッピングを設定します。トランスフォーメーション間の不要なリンクを削除して、データの移動量を最小限に抑えます。マッピング内のトランスフォーメーションを最適化します。その際は、マッピング設計のベストプラクティスに従うようにしてください。

## 区切りフラットファイルソースの最適化

ソースが区切りフラットファイルである場合は、区切り文字を指定してソースファイルのデータの列を区切ります。また、エスケープ文字を指定します。

区切り文字の前にエスケープ文字を挿入すると、データ統合は区切り文字を通常の文字として読み取ります。ソースのフラットファイルに引用符やエスケープ文字が含まれないようにすることで、パフォーマンスを向上させることができます。

## データプレビューの最適化

データプレビューのパフォーマンスは、プレビューする行数に応じて変化します。デフォルトでは、マッピングは 100 行をプレビューします。

1 GB を超えるソースの場合は、**【ソース全体の読み取り】** および **【アップストリームプレビューの有効化】** の選択を解除します。

## フィルタの最適化

データフローの初期の部分でフィルタを使用して、マッピングのパフォーマンスを向上させます。

データをフィルタするには、次のいずれかのトランスフォーメーションを使用します。

- ソーストランスフォーメーション。ソーストランスフォーメーションにより、データベースまたはデータウェアハウスソースからの行をフィルタリングします。
- フィルタトランスフォーメーション。フィルタトランスフォーメーションにより、マッピング内のデータをフィルタリングします。Filter トランスフォーメーションは、すべての種類のソースからの行をフィルタリングします。

マッピングからのレコードにフィルタを適用する場合、データフローの早めの段階でフィルタリングを行うことによって効率を上げることができます。ソーストランスフォーメーションのフィルタを使用して、ソースから行を削除します。ソーストランスフォーメーションにより、データベースまたはデータウェアハウスソースから抽出される行セットを制限します。

ソーストランスフォーメーションのフィルタを使用できない場合は、フィルタトランスフォーメーションを使用し、ソーストランスフォーメーションに可能な限り近い部分にこのフィルタトランスフォーメーションを移動して、データフローの初期の部分にある不必要なデータを削除します。Filter トランスフォーメーションは、ターゲットに送信される行セットを制限します。

フィルタ条件では、複雑な式を使用しないようにします。フィルタトランスフォーメーションを最適化するには、フィルタ条件で整数の簡易式または真偽式を使用します。

# データ型変換の最適化

不要なデータ型変換を除去することで、パフォーマンスが向上します。

例えば、Integer カラムから Decimal カラムへデータを移動した後に再び Integer カラムにそのデータを戻すマッピングの場合は、不要なデータ型変換によってパフォーマンスが低下します。不要なデータ型変換は、マッピングから可能な限り除去するようにします。

パフォーマンスを向上させるには、次のようなデータ型変換を使用します。

- ルックアップトランスフォーメーションとフィルタトランスフォーメーションを使用して比較を行う場合は、Integer 以外のデータ型についても Integer 値を使用します。例えば、多くのデータベースでは米国の郵便番号情報が Char データ型または Varchar データ型として格納されています。郵便番号のデータを Integer データ型に変換した場合、ルックアップデータベースでは郵便番号 94303-1234 が 943031234 という値で格納されます。これにより、郵便番号に基づいたルックアップの比較条件の処理が高速化します。
- フィールドからフィールドへの変換を介してソースの日付を文字列に変換し、マッピングのパフォーマンスを向上させます。ターゲットのフィールドは、文字列のままにするか Date/Time フィールドに変更することもできます。

## 式の最適化

トランスフォーメーションで使用される式を最適化できます。可能な場合は、パフォーマンスを低下させている式を見つけて簡素化します。

パフォーマンスを低下させている式を見つけるため、次のタスクを実行します。

1. マッピングから 1 つずつ式を削除します。
2. マッピングを実行して、トランスフォーメーションを使用せずにマッピングを実行したときの時間を測定します。マッピングの実行時間が著しく異なる場合は、パフォーマンスを低下させている式を最適化する方法を確認します。

## 共通ロジックの排除

マッピングで同じ処理が複数の箇所で実行される場合は、その処理をマッピングの初期の部分に移すことで、マッピングによる処理の実行回数を減らします。

例えば、マッピングに 5 つのターゲットテーブルがあるとします。各ターゲットには、社会保障番号のルックアップが必要です。この場合、ルックアップを 5 回実行する代わりに、データフローが分かれる前のマッピングに Lookup トランスフォーメーションを配置します。次に、ルックアップの結果を 5 つのターゲットすべてに渡します。

## 集計関数呼び出しの最小化

式を記述するときは、可能な限り少ない集計関数の呼び出しを使用します。

集計関数が呼び出されるたびに、マッピングはデータの検索とグループ化を行います。例えば、次の式では、データ統合は最初に COLUMN\_A を読み取ってその合計を算出します。次に、COLUMN\_B を読み取ってその合計を算出し、最後に 2 つの合計の和を算出します。

`SUM(COLUMN_A) + SUM(COLUMN_B)`

次の例のように集計関数呼び出しを減らした場合、データ統合は COLUMN\_A を COLUMN\_B に加えた後にその合計を算出します。

```
SUM(COLUMN_A + COLUMN_B)
```

## ローカル変数での共通の式の置き換え

1 つのトランスフォーメーションで同じ式が複数回使用されている場合は、その式をローカル変数にすることができます。

ローカル変数はトランスフォーメーション内のみで使用できます。ただし、変数の計算が一度で済むため、パフォーマンスを向上させることができます。

## 文字列演算の代わりに数値演算の選択

マッピングは、文字列演算よりも数値演算を高速に処理します。

例えば、2 つのフィールド（EMPLOYEE\_NAME と EMPLOYEE\_ID）で大量のデータを検索する場合は、EMPLOYEE\_ID にルックアップを設定することでパフォーマンスが向上します。

## LOOKUP の代わりに DECODE の選択

LOOKUP 関数を使用すると、マッピングはデータベース内のテーブルのルックアップを行います。DECODE 関数を使用すると、ルックアップ値が式自体に取り込まれるため、マッピングは個々のテーブルのルックアップを行わなくなります。ルックアップする値が変化せず、その数が少ない場合は、DECODE を使用するとパフォーマンスが向上します。

## 関数の代わりに演算子の使用

マッピングは、関数を使用して記述された式よりも演算子を使用して記述された式を高速に読み取ります。そのため、可能な限り演算子を使用して式を記述するようにします。

たとえば、CONCAT 関数をネストした次のような式があるとします。

```
CONCAT( CONCAT( CUSTOMERS.FIRST_NAME, ' ' ) CUSTOMERS.LAST_NAME)
```

|| 演算子を使用すると、上記の式を次のように表すことができます。

```
CUSTOMERS.FIRST_NAME || ' ' || CUSTOMERS.LAST_NAME
```

## IIF 関数の最適化

IIF 関数は値およびアクションを返すことができるため、式をより簡潔にすることができます。

例えば、FLG\_A、FLG\_B、FLG\_C という 3 つの Y/N フラグを含むソースがあり、それぞれのフラグの値に基づいた値を返す必要があるとします。次の式を使用します。

```
IIF( FLG_A = 'Y' and FLG_B = 'Y' AND FLG_C = 'Y', VAL_A + VAL_B + VAL_C,
    IIF( FLG_A = 'Y' and FLG_B = 'Y' AND FLG_C = 'N', VAL_A + VAL_B ,
        IIF( FLG_A = 'Y' and FLG_B = 'N' AND FLG_C = 'Y', VAL_A + VAL_C,
            IIF( FLG_A = 'Y' and FLG_B = 'N' AND FLG_C = 'N', VAL_A ,
                IIF( FLG_A = 'N' and FLG_B = 'Y' AND FLG_C = 'Y', VAL_B + VAL_C,
                    IIF( FLG_A = 'N' and FLG_B = 'Y' AND FLG_C = 'N', VAL_B ,
                        IIF( FLG_A = 'N' and FLG_B = 'N' AND FLG_C = 'Y', VAL_C,
                            IIF( FLG_A = 'N' and FLG_B = 'N' AND FLG_C = 'N', 0.0,
                                )))))))
```

この式では、8 個の IIF、16 個の AND、および少なくとも 24 回の比較が必要です。

IIF 関数を利用すると、上記の式を次のように表すことができます。

```
IIF(FLG_A='Y', VAL_A, 0.0)+ IIF(FLG_B='Y', VAL_B, 0.0)+ IIF(FLG_C='Y', VAL_C, 0.0)
```

この式で使用する IIF は 3 個、比較は 2 個、加算は 2 個となるため、タスクの実行が高速化されます。

## ウィンドウ関数の最適化

ウィンドウ関数は小さいフレームをより高速に処理します。フレームのサイズを小さくするには、フレームオフセット間の行数を減らし、先行するすべての行または後続のすべての行を含めないようにします。

適切に分散されたパーティションキーを使用して、同様のサイズのパーティションを作成します。

## 式の評価

パフォーマンスを低下させている式がどの式であるか不明な場合は、式のパフォーマンスを評価して問題を特定します。

式のパフォーマンスを評価するには、次の手順を実行します。

1. 元の式を使用してマッピングの実行時間を計測します。
2. マッピングをコピーして、複合式の半分を定数に置き換えます。
3. 編集したマッピングを実行して時間を計測します。
4. マッピングのコピーをもう 1 つ作成し、複合式の残りの半分を定数で置き換えます。
5. 編集したマッピングを実行して時間を計測します。

## アグリゲータトランスフォーメーションの最適化

アグリゲータトランスフォーメーションはデータを処理する前にグループ化を行うため、パフォーマンスが低下しやすくなります。そのため、アグリゲータトランスフォーメーションには中間のグループ結果を格納するための追加メモリが必要となります。

最適なパフォーマンスを得るためには、アグリゲータトランスフォーメーションをソーストランスフォーメーションのできるだけ近くに配置します。

Aggregator トランスフォーメーションのパフォーマンスを最適化するには、以下のガイドラインに従います。

- 単純なカラム別にグループ化する。
- ソート済み入力を使用する。
- 集計する前にデータをフィルタリングする。
- フィールド接続を制限する。

## 単純なカラム別のグループ化

単純なカラム別のグループ化を行う際は、Aggregator トランスフォーメーションを最適化することができます。可能な場合には、GROUP BY で使用するカラムでは、文字列や日付ではなく数値を使用します。

Aggregator の式が複雑にならないようにします。



## ソート済み入力の使用

マッピングのパフォーマンスを向上させるには、アグリゲータトランスフォーメーションのデータをソートします。データのソートには、Sorted Input オプションを使用します。

Sorted Input オプションは、集計キャッシュの使用頻度を低くします。[ソート済み入力] オプションを使用すると、データ統合ではすべてのデータがグループごとにソートされているとみなされます。データ統合は1つのグループに対する行を読み取るときに集計計算を実行します。必要に応じて、データ統合はメモリにグループ情報を格納します。

[ソート済み入力] オプションにより、タスクの実行中にキャッシュに格納されるデータ量を減らして、パフォーマンスを向上させます。ソート済みのデータをアグリゲータトランスフォーメーションに渡すには、ソーストランスフォーメーションのソースフィルタまたはソータートランスフォーメーションのいずれかでソートされた入力を使用します。

複数のパーティションを含むマッピングで入力をソートすると、パフォーマンスが向上します。

## 集計前のデータのフィルタリング

集計する前にデータをフィルタリングします。マッピングで Filter トランスフォーメーションを使用する場合、そのあとに Aggregator トランスフォーメーションを置いて、不要な集計を減らしてください。

## 接続されたフィールドの制限

接続された入出力フィールドまたは出力フィールドの数を制限して、アグリゲータトランスフォーメーションがデータキャッシュに格納するデータの量を減らします。

# 階層プロセッサトランスフォーメーションの最適化

階層プロセッサトランスフォーメーションは、階層データの複雑さに基づいてパフォーマンスを低下させる可能性があります。

階層プロセッサトランスフォーメーションのパフォーマンスを最適化するには、次のガイドラインを使用します。

- 階層プロセッサトランスフォーメーションに渡す前にソーストランスフォーメーションの階層データを読み取る場合は、インテリジェント構造モデルの代わりに可能な限り JSON 形式を使用するようにします。ただし、1 MB を超えるファイルにはインテリジェント構造モデルを使用してください。
- 100 MB を超える分割不可能なファイルは避けてください。代わりに、いくつかの小さな JSON ファイルを処理します。
- ネストされた階層の数を減らします。マッピングは、5 つのレベルを持つ階層よりも 3 つのレベルを持つ階層を高速に処理します。

# ジョイナトランスフォーメーションの最適化

ジョイナトランスフォーメーションは、実行時に中間結果を格納するための追加領域が必要となるため、パフォーマンスが低下することがあります。

ジョイナトランスフォーメーションのパフォーマンスを向上させるには、次のガイドラインを使用してください。

## 重複キー値が少ないマスターグループをソースとして指定する。

データ統合は、ソート済みのジョイナトランスフォーメーションを処理するときに、100 個の一意のキーに対する行を一度にキャッシュに格納します。マスターグループに同じキー値を持つ多数の行が含まれている場合、データ統合はより多くの行をキャッシュに格納する必要があるため、パフォーマンスが低下することがあります。

## 行数が少ないマスターグループをソースとして指定する。

ジョイナトランスフォーメーションは詳細グループの各行をマスターグループと比較します。マスター内の行が少ない場合は結合のために比較が繰り返される回数が少なくなるため、結合プロセスが高速になります。

## 可能な場合は、データベースまたはデータウェアハウスで結合を実行する。

データベース内で結合を実行すると、マッピング内で実行する場合よりも処理が高速になります。パフォーマンスは、使用するデータベース結合の種類によっても変わってきます。ノーマルジョインは、アウトジョインよりも高速で、結果的にレコード数が少なくて済みます。場合によって、たとえば 2 つの異なるデータベースまたはフラットファイルシステムとテーブルを結合する場合は、これが不可能なこともあります。

## 可能な場合は、ソート済みデータを結合する。

マッピングのパフォーマンスを向上させるには、ソート済みの入力を使用するようにジョイナトランスフォーメーションを設定します。ジョイナトランスフォーメーションでソート済みデータを使用するように設定すると、データ統合ではディスクの入出力が最小化され、パフォーマンスが向上します。これにより、大量のデータセットを処理する場合にパフォーマンスを大幅に向上させることができます。未ソートのジョイナトランスフォーメーションの場合は、行の少ないソースをマスターソースとして指定します。

## 最大のデータセットを最後に結合する。

複数のジョイナトランスフォーメーションを含むマッピングでは、最もダウンストリームのトランスフォーメーションで最大のデータセットを結合します。

## ブロードキャスト結合のしきい値を設定する。

詳細モードのマッピングは、Spark セッションプロパティ `spark.sql.autoBroadcastJoinThreshold` に設定された値よりも小さいデータセットに対してブロードキャスト結合を実行します。このマッピングにより、すべての詳細クラスターノードですべての Spark エグゼキュータにデータセットがブロードキャストされるため、シャッフルオーバーヘッドが削減され、パフォーマンスが向上します。

CLAIRE チューニングを実行して、ブロードキャスト結合のしきい値の推奨事項を取得します。

# ルックアップトランスフォーメーションの最適化

ルックアップテーブルがソーステーブルと同じデータベース上にあり、キャッシュが実行できない場合は、ルックアップトランスフォーメーションを使用する代わりに、ソースデータベース内のテーブルを結合します。

ルックアップトランスフォーメーションを使用する場合は、次のタスクを実行してパフォーマンスを向上させることができます。

- Lookup テーブルをキャッシュする。
- Lookup 条件を最適化する。
- Lookup の行をフィルタする。
- Lookup テーブルをインデックスに入れる。
- 複数の Lookup を最適化する。

## ルックアップテーブルのキャッシュ

マッピングにルックアップトランスフォーメーションが含まれている場合は、ルックアップキャッシュを有効にします。

キャッシュを有効にすると、データ統合はルックアップテーブルをキャッシュに格納して、タスクの実行中にルックアップキャッシュを参照します。このオプションが有効ではない場合、データ統合は行ごとにルックアップテーブルを参照します。

ルックアップテーブルをキャッシュに格納するかどうかに関わらず、ルックアップクエリの結果および処理は同じです。ただし、ルックアップキャッシュを使用するとマッピングのパフォーマンスを向上させることができます。通常、必要なサイズが 300MB 未満のルックアップテーブルは、キャッシュに入れます。

ルックアップキャッシュを有効にした場合は、次のタスクを実行するとパフォーマンスが向上します。

- 適切なキャッシュタイプを使用する。
- コンカレントキャッシュを有効にする。
- ルックアップ条件の一致を最適化する。
- キャッシュに入れるレコード数を減らす。
- ORDER BY 文を上書きする。
- メモリの多いマシンを使用する。

## キャッシュタイプ

パフォーマンスを向上させるために使用するキャッシュタイプを検討します。

次のキャッシュタイプを使用できます。

### 永続キャッシュ

キャッシュファイルを保存して再使用するには、永続キャッシュを使用するようにトランスフォーメーションを設定します。永続キャッシュは、タスク間でルックアップテーブルが変化しないことがわかっている場合に使用します。永続キャッシュを使用すると、データ統合はデータベースではなくキャッシュファイルからメモリキャッシュを作成するため、パフォーマンスが向上します。

### 動的キャッシュ

ルックアップキャッシュがターゲットと同期し続けるようにするには、動的ルックアップキャッシュを使用します。キャッシュが静的である場合、マッピングタスクの実行時にルックアップキャッシュ内のデータは変更されません。タスクがキャッシュを複数回使用する場合、タスクは同じデータを使用します。キ

キャッシュが動的である場合、タスクはタスク内のアクションに基づいてキャッシュを更新します。そのため、タスクがルックアップを複数回使用した場合、ダウストリームトランスフォーメーションは更新されたデータを使用することになります。

## コンカレントキャッシュの有効化

データ統合がルックアップトランスフォーメーションを含むマッピングを実行すると、キャッシュされたルックアップトランスフォーメーションのデータの最初の行を処理するときにメモリ内にキャッシュが作成されます。マッピングに複数のルックアップトランスフォーメーションがある場合、データ統合は、データの最初の行がルックアップトランスフォーメーションによって処理されたときに、キャッシュを順番に作成します。これにより、Lookup トランスフォーメーションの処理は遅くなります。

コンカレントキャッシュを有効にすると、パフォーマンスを向上させることができます。追加のコンカレントパイプラインの数が 1 つ以上に設定されている場合、データ統合は順番に作成する代わりに同時にキャッシュを作成します。アグリゲータ、ジョイナ、またはソータトランスフォーメーションなど、完了までに時間のかかるアクティブなトランスフォーメーションがタスクに多数含まれている場合は、パフォーマンスが大幅に向上します。複数のコンカレントパイプラインを有効にした場合、データ統合は、アクティブなタスクの実行が完了した後にキャッシュをすぐに作成します。パイプライン内の他の Lookup トランスフォーメーションも、同時にキャッシュを作成します。

## ルックアップ条件の一致の最適化

ルックアップトランスフォーメーションがルックアップキャッシュデータとルックアップ条件を照合したときに、最初に一致した値と最後に一致した値を判断するために、ルックアップトランスフォーメーションはデータをソートし、順序を入れ替えます。

Lookup 条件に一致する値を何か返すように、トランスフォーメーションを設定できます。Lookup トランスフォーメーションが一致する値を何か返すよう設定すると、トランスフォーメーションは Lookup 条件に一致する最初の値を返します。最初に一致した値または最後に一致した値を返すようにトランスフォーメーションを設定した場合のように、すべてのフィールドがインデックス処理されることはありません。任意の一致する値を使用すると、トランスフォーメーションによって一部のフィールドのみがインデックス処理されるため、パフォーマンスが向上する可能性があります。

## キャッシュされる行の削減

キャッシュに入れるレコード数を減らして、パフォーマンスを向上させることができます。Lookup SQL Override オプションを使用すると、デフォルトの SQL 文に WHERE 句を追加できます。

## ORDER BY 文の上書き

デフォルトでは、データ統合は、キャッシュされたルックアップに対して ORDER BY 文を生成します。ORDER BY 文にはすべてのルックアップフィールドが含まれています。

パフォーマンスを向上させるには、デフォルトの ORDER BY 文を使用する代わりに、カラムの数が少ない上書き ORDER BY を入力します。

上書きで ORDER BY 文を入力した場合でも、データ統合は常に ORDER BY 文を生成します。上書き ORDER BY の後に 2 個のダッシュ (--) を挿入することで、生成した ORDER BY 文を抑止できます。

例えば、ルックアップトランスフォーメーションが次のルックアップ条件を使用するとします。

```
ITEM_ID = IN_ITEM_ID  
PRICE <= IN_PRICE
```

ルックアップトランスフォーメーションには、マッピングで使用される 3 つのルックアップフィールド (ITEM\_ID、ITEM\_NAME、および PRICE) が含まれます。ORDER BY 文を入力する場合は、フィールドを入力

したときと同じ順序でルックアップ条件にカラムを入力します。すべてのデータベース予約語を引用符で囲みます。

ルックアップ SQL オーバーライドで次のルックアップクエリを入力します。

```
SELECT ITEMS_DIM.ITEM_NAME, ITEMS_DIM.PRICE, ITEMS_DIM.ITEM_ID FROM ITEMS_DIM ORDER BY ITEMS_DIM.ITEM_ID, ITEMS_DIM.PRICE --
```

## メモリの多いマシンの使用

パフォーマンスを向上させるには、大量のメモリを備えた Secure Agent マシンでマッピングを実行します。マシンに負荷がかからない範囲で、インデックスキャッシュサイズおよびデータキャッシュサイズを可能な限り増やします。

Secure Agent マシンに十分なメモリがある場合は、ディスクへのページングを行わなくてもすべてのデータをメモリ内に保存できるようにキャッシュを増やします。

## ルックアップ条件の最適化

複数のルックアップ条件を含める場合は、ルックアップのパフォーマンスを向上させるために、最適な順序で条件を配置します。

次の順序を使用します。

1. 等しい (=)
2. より小さい (<)、より大きい (>)、より小さいまたは等しい (<=)、より大きいまたは等しい (>=)
3. 等しくない (!=)

## ルックアップ行のフィルタリング

ルックアップキャッシュ作成時にソースから取得されるルックアップ行の数を減らすには、フィルタ条件を作成します。

## ルックアップテーブルのインデックス処理

データ統合は、ルックアップ条件カラムで値のクエリ、ソート、および比較を行う必要があります。インデックスには、Lookup 条件で使用する各カラムを含めなければなりません。

次の種類の Lookup において、パフォーマンスを向上させることができます。

- キャッシュを使用するルックアップ。ルックアップ ORDER BY 文でカラムをインデックス処理します。セッションログには ORDER BY 文が含まれます。
- キャッシュを使用しないルックアップ。ルックアップ条件内のカラムをインデックス処理します。データ統合は、ルックアップトランスフォーメーションに渡される各行に対して SELECT 文を発行します。

## 複数のルックアップの最適化

マッピングに複数のルックアップが含まれている場合、キャッシングが有効になっていてヒープメモリが十分にあっても、ルックアップによりパフォーマンスが低下することがあります。一番多くのデータを問い合わせるルックアップトランスフォーメーションを探して、全体のパフォーマンスを向上させることができます。

# 機械学習トランスフォーメーションの最適化

機械学習トランスフォーメーションを最適化するには、複数の要求を 1 つの要求に結合し、トランスフォーメーションによって機械学習モデルに送信されるように一括要求を設定します。

## ノーマライザトランスフォーメーションの最適化

ノーマライザトランスフォーメーションでは行が生成されます。パフォーマンスを最適化するには、ターゲットに可能な限り近い部分にノーマライザトランスフォーメーションを配置します。

## ルータートランスフォーメーションの最適化

ルータートランスフォーメーションが詳細モードのマッピングで多くの出力グループを処理する場合は、マッピングタスクで Spark セッションプロパティ `infaspark.sql.forcePersist=true` を設定することで、データを永続化できます。

データを永続化すると、ルータートランスフォーメーションで出力グループごとに対する読み取り操作が発生しなくなります。

## シーケンスジェネレータトランスフォーメーションの最適化

シーケンスジェネレータトランスフォーメーションを最適化するには、再利用可能なシーケンスジェネレータを作成して、複数のマッピング内でシーケンスジェネレータを同時に使用します。また、一度にキャッシュする値の数を設定します。

キャッシュする値の数が少なくなりすぎないようにしてください。値の数は、1,000 よりも大きい値に設定する必要があります。

値をキャッシュする必要がない場合は、キャッシュする値の数を 0 に設定します。キャッシュを使用しないシーケンスジェネレータトランスフォーメーションは、キャッシュを必要とするトランスフォーメーションよりも高速になります。

シーケンスジェネレータトランスフォーメーションで `CURRVAL` フィールドを接続すると、データ統合は各ブロックで 1 行ずつ処理します。マッピングで `NEXTVAL` フィールドのみを接続することで、パフォーマンスを最適化できます。

# ソータートランスフォーメーションの最適化

ソータートランスフォーメーションを最適化するには、データをソートするために必要なメモリを割り当て、トランスフォーメーション内の各パーティションに異なる作業ディレクトリを指定します。

## メモリの割り当て

最適なパフォーマンスを得るには、Secure Agent マシンで利用可能な物理 RAM 量以下の値をソーターキャッシュサイズに設定します。

ソータートランスフォーメーションを使用してデータをソートするには、最低 16MB の物理メモリを割り当てます。デフォルトでは、ソーターキャッシュサイズは 16,777,216 バイトに設定されます。データ統合がデータのソートを行うために必要なメモリを割り当てられない場合、マッピングタスクは失敗します。

入力されるデータの量がソーターキャッシュサイズよりも大きい場合、データ統合はデータを一時的にソータートランスフォーメーションの作業ディレクトリに保存します。データを作業ディレクトリに保存する場合、データ統合には、最低でも入力されるデータの量の 2 倍のディスク領域が必要となります。入力されるデータの量がソーターキャッシュサイズよりもはるかに大きい場合、データ統合には、ワークディレクトリで利用可能なディスク領域の 2 倍の量よりもさらに多くの領域が必要となる可能性があります。

**注:** セッションログには、ソータートランスフォーメーションに対する入力行数および入力データのサイズが含まれます。例えば、データ統合がソータートランスフォーメーションを処理した場合は、次のようなメッセージが表示されます。

```
SORT_40422 End of output from Sorter Transformation [srt_1_Billion_FF]. Processed 999999999 rows (866325637228 input bytes; 868929593344 temp I/O bytes)
```

このメッセージの「999999999」は入力行の数を表し、「866325637228」は入力行のサイズを表します。

## 作業ディレクトリの指定

データ統合は、データをソートするときに一時ファイルを作成します。一時ファイルは、ワークディレクトリに保存されます。Secure Agent マシンの任意のディレクトリを作業ディレクトリとして使用するよう指定できます。

デフォルトでは、データ統合は \$PMTempDir サービスプロセス変数に指定されている値を使用します。ソータートランスフォーメーションを使用してマッピングをパーティション化する場合、パイプライン内のパーティションごとに異なる作業ディレクトリを指定できます。マッピングのパフォーマンスを向上させるには、Secure Agent マシン上の物理的に別のディスクに作業ディレクトリを指定します。

ソータートランスフォーメーションの作業ディレクトリを指定するには、ソータートランスフォーメーションの詳細プロパティ **【作業ディレクトリ】** を使用します。



## 第 5 章

# マッピングタスクの最適化

ボトルネックを特定して除去することで、マッピングタスクのパフォーマンスを最適化できます。

キャッシュのサイズ、バッファのメモリ容量、およびコミット間隔の不足によって、マッピングタスクのボトルネックが発生することがあります。

## マッピングタスクのボトルネックの特定

マッピングタスクのボトルネックを特定するには、パフォーマンスの詳細を分析します。パフォーマンスの詳細には、入力行、出力行およびエラー行の数など、各トランスフォーメーションに関する情報が表示されます。

## マッピングタスクのボトルネックの除去

マッピングタスクのボトルネックを除去するには、マッピングタスクを最適化します。

### バッファメモリ

データ統合では、タスクの初期化時に、ソースデータとターゲットデータを保持するためのメモリブロックが割り当てられます。

データ統合は、ソースパーティションおよびターゲットパーティションごとに少なくとも 2 つのブロックを割り当てます。多数のソースやターゲットを使用するマッピングタスクでは、追加のメモリブロックが必要になる場合があります。データ統合がデータを保持するために必要なメモリを割り当てられない場合、タスクは失敗します。

バッファメモリの容量を設定するか、実行時にバッファ設定を計算するようにデータ統合を設定することもできます。

使用可能なメモリブロック数を増やすには、次のようなマッピングタスクのプロパティを調整します。

- **DTM バッファサイズ。** マッピングタスクの DTM バッファサイズの詳細セッションプロパティを増やします。
- **バッファブロックサイズ。** マッピングタスクのバッファブロックサイズの詳細セッションプロパティを減らします。



**注:** データのパーティション化が有効な場合の DTM バッファサイズは、すべてのパーティションに割り当てられたすべてのメモリバッファプールの合計サイズになります。 $n$  個のパーティションを含むタスクの場合は、1 つのパーティションだけを持つタスクの値に対して少なくとも  $n$  倍の DTM バッファサイズを設定します。

## DTM バッファサイズの拡大

DTM バッファサイズの設定により、データ統合が DTM バッファメモリとして使用するメモリの量を指定します。DTM バッファメモリを増やすと、データ統合で作成されるバッファブロックが増加するため、一時的な速度低下時のパフォーマンスが向上します。

DTM バッファメモリの割り当てを増やすと、一般的には、最初にパフォーマンスが向上した後に安定します。パフォーマンスが大きく向上しない場合は、DTM バッファメモリの割り当てがマッピングのパフォーマンスに影響を及ぼしていないと判断できます。

DTM バッファサイズを増やすには、タスクを開いて、**[DTM バッファサイズ]** の詳細セッションプロパティを編集します。バッファブロックサイズの倍数で DTM バッファサイズを増やします。

## バッファブロックサイズの最適化

Secure Agent マシンの物理的なメモリが制限されており、マッピングに多くのソース、ターゲット、パーティションがあるときは、バッファサイズを減らす必要がある場合もあります。

非常に大きなデータ行を処理する場合は、バッファブロックサイズを増やすとパフォーマンスが向上します。行のおおよそのサイズがわからない場合は、次の手順を実行して行サイズを判断します。

1. **[エクスプローラ]** ページでマッピングを開きます。
2. ターゲットトランスフォーメーションを開きます。
3. **[ターゲットフィールド]** タブをクリックします。
4. ターゲット内のすべてのカラムに精度を追加します。
5. マッピング内に複数のターゲットがある場合は、追加するターゲットごとに手順 2-4 を繰り返し、各ターゲットの精度を計算します。
6. マッピング内の各ソース定義に対して、手順 2-5 を繰り返します。
7. ソースおよびターゲットの精度すべての中から最大の精度を選択してバッファサイズの合計の精度を求めてください。

合計の精度は、データの最大のレコード 1 つを移動するのに必要な合計のバイト数を表します。例えば、合計の精度が 33,000 である場合、データ統合でその行を移動するには 33,000 バイトのバッファブロックが必要になります。バッファブロックサイズが 64,000 バイトのみである場合、データ統合で一度に複数の行を移動することはできません。

バッファブロックサイズを設定するには、タスクを開いて、**[デフォルトのバッファブロックサイズ]** 詳細セッションプロパティを編集します。

DTM バッファメモリの割り当てと同様に、バッファブロックサイズを増やすことでパフォーマンスが向上します。パフォーマンスが向上しない場合、バッファブロックサイズはタスクのパフォーマンスの要因ではないと判断できます。

## キャッシュ

データ統合は、XML ターゲットと、アグリゲータ、ランク、ルックアップ、およびジョイナトランスフォーメーションに対してインデックスとデータキャッシュを使用します。

データ統合は、パイプラインに返す前に、変換されたデータをデータキャッシュに格納します。データ統合はグループ情報をインデックスキャッシュに格納します。また、データ統合はキャッシュを使用して、ソータートランスフォーメーションのデータを格納します。

キャッシュメモリの容量を設定するには、キャッシュサイズを指定します。格納するデータの容量よりも割り当てられたキャッシュが少ない場合、データ統合はタスクデータの処理を行うときに一時ディスクファイル（キャッシュファイル）内にデータを格納します。そのため、データ統合が一時ファイルにページングを行うたびにパフォーマンスが低下します。

キャッシュを最適化するには、次のタスクを実行します。

- 接続された入出力フィールドと出力専用フィールドの数を制限する。
- キャッシュサイズを増やす。

## 接続されたフィールドの制限

データキャッシュを使用するトランスフォーメーションで、接続された入出力フィールドと出力専用フィールドを制限します。入出力フィールドまたは出力フィールドの接続数を制限することで、トランスフォーメーションがデータキャッシュに格納するデータ量を減らします。

## キャッシュサイズの増量

キャッシュサイズを設定して、トランスフォーメーションを処理するために割り当てられるメモリの量を指定します。設定するメモリの量は、使用するメモリキャッシュおよびディスクキャッシュのサイズによって異なります。

キャッシュサイズの容量が十分ではない場合、データ統合はメモリ内のトランスフォーメーションの一部のみを処理し、残りのトランスフォーメーションを処理するために情報をキャッシュファイルにページングします。そのため、データ統合がキャッシュファイルにページングを行うたびに、パフォーマンスが低下します。

マッピング内にキャッシュを使用するトランスフォーメーションがあり、十分にメモリのあるマシンでタスクを実行する場合は、キャッシュサイズを増やして、トランスフォーメーションがメモリで処理されるようにします。

## 詳細ログ

マッピングタスクは、標準実行モードまたは詳細実行モードで実行できます。タスクを詳細実行モードで実行すると、マッピングにより、トラブルシューティングに使用できる追加データがログに生成されます。

詳細実行モードは、トラブルシューティングを行う場合にのみ使用します。詳細実行モードで生成されるデータ量は、パフォーマンスに影響を及ぼします。

## 詳細モードのベストプラクティス

詳細モードのマッピングに基づいてマッピングタスクを作成する場合は、CLAIRE チューニングを実行して、ジョブのパフォーマンスを向上させるための推奨事項を取得します。CLAIRE によって推奨される設定には、Spark ドライバ、Spark エグゼキュータ、およびその他の Spark セッションプロパティなどがあります。

## 第 6 章

# 詳細クラスタの最適化

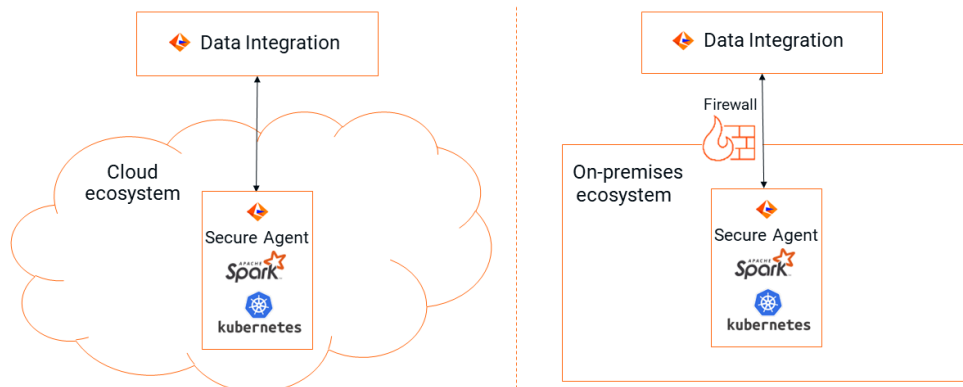
ワークロードを処理するために必要なリソースをプロビジョニングすることで、詳細クラスタのパフォーマンスを最適化できます。

## 詳細クラスタのコンポーネント

データ統合は、クラスタのタイプに基づいて詳細クラスタのコンポーネントとデータのやり取りを行います。

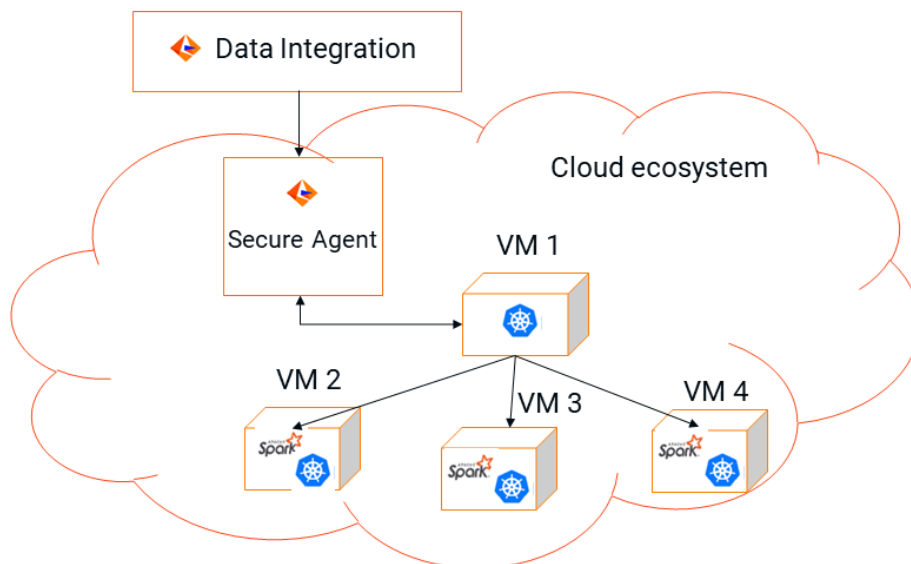
### ローカルクラスタ

次の図に、データ統合がクラウドとオンプレミス両方のエコシステムのローカルクラスタ内の Secure Agent および詳細クラスタのコンポーネントとデータのやり取りを行う方法を示します。



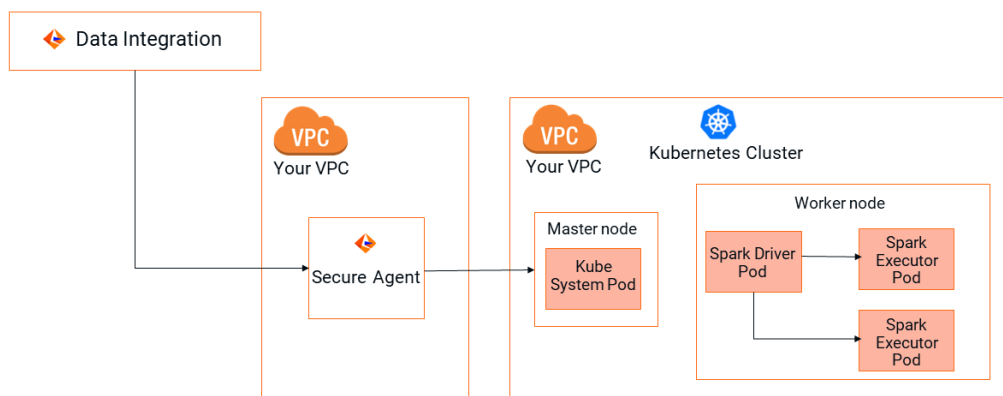
## フルマネージドクラスタ

次の図に、データ統合がフルマネージドクラスタ内の Secure Agent および詳細クラスタのコンポーネントとデータのやり取りを行う方法を示します。



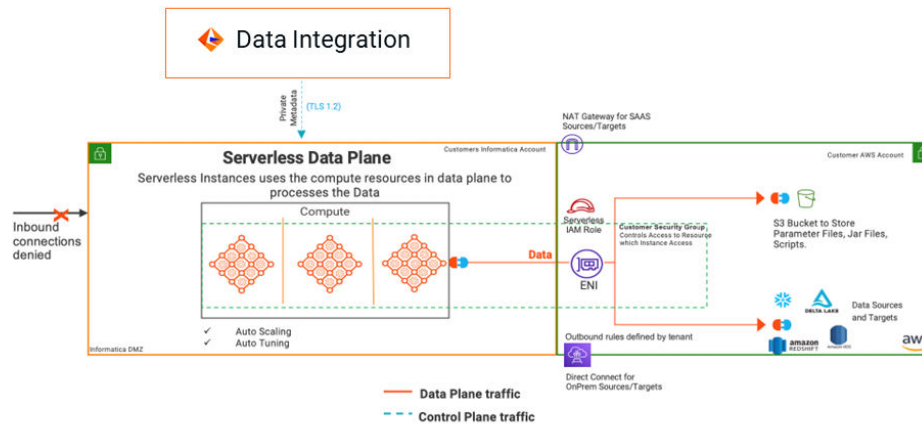
## セルフサービスクラスタ

次の図に、データ統合がセルフサービスクラスタ内の Secure Agent および詳細クラスタのコンポーネントとデータのやり取りを行う方法を示します。



## サーバーレスランタイム環境の詳細クラスタ

次の図に、データ統合がサーバーレスランタイム環境で詳細クラスタのコンポーネントとデータのやり取りを行う方法を示します。



## ベストプラクティス

詳細設定を作成する場合は、ベストプラクティスに従って詳細クラスタのパフォーマンスを最適化します。

次のベストプラクティスを考慮してください。

### ストレージの自動スケーリングを有効にする。

ストレージの自動スケーリングを使用して、ジョブの処理に使用できるディスク領域を動的に変更します。ジョブには、ジョブ内のデータロジックとデータボリュームに基づいたディスク領域が必要となります。

### ワーカーノードの自動スケーリングを有効にする。

ワーカーノードの自動スケーリングを使用して、ジョブの処理に使用できるノードの数を動的に変更します。

自動スケーリングの詳細については、次の Informatica ブログを参照してください:

[Cloud Data Integration Elastic - Understanding Auto Scaling](#)。

### スポットインスタンスを使用する。

スポットインスタンスは、オンデマンドインスタンスと同じパフォーマンスを低価格で提供します。ただし、スポットインスタンスは常に利用できるとは限りません。中断の頻度が 5%未満のスポットインスタンスを使用する。スポットインスタンスとその中断頻度のリストについては、AWS の [Spot Instance advisor](#) を参照してください。

開発環境と QA 環境では、スポットインスタンスを使用することで、内部テストとデバッグ中のコストを節約できます。厳格な SLA を持つジョブのスポットインスタンスの使用は避けるようにしてください。

# 詳細クラスターノードの最適化

各ノードタイプに必要なリソースをプロビジョニングすることで、マスターノードとワーカーノードを最適化できます。

## マスターノードの最適化

詳細クラスター内のワーカーノードを管理するために必要なリソースを持つマスターノードをプロビジョニングします。

次の表に、ワーカーノードの数に基づいたマスターノードの構成を示します。

| ワーカーノードの数 | マスターノードの構成            |
|-----------|-----------------------|
| 1-10      | 4 個の CPU と 8 GB のメモリ  |
| 11-100    | 8 個の CPU と 32 GB のメモリ |

## ワーカーノードの最適化

ジョブを処理するために必要なリソースをワーカーノードにプロビジョニングします。

次の表に、詳細クラスターのタイプに基づいたワーカーノードの構成を示します。

| 詳細クラスターのタイプ  | ワーカーノードの構成  |
|--------------|---|
| フルマネージドクラスター | 少なくとも 8 個の CPU と 32 GB のメモリ。<br>パフォーマンスを向上させるには、16 個の CPU と 64 GB のメモリを使用します。 |
| ローカルクラスター    | 少なくとも 4 個の CPU と 16 GB のメモリ。<br>パフォーマンスを向上させるには、8 個の CPU と 32 GB のメモリを使用します。  |

プロビジョニングするワーカーノードの数は、SLA によって異なります。

# インスタンスタイプの最適化

詳細クラスターで処理するデータロジックに基づいて、ワーカーノードのインスタンスタイプを選択します。

## GPU インスタンスの使用

GPU インスタンスにより、パフォーマンスが 5 倍向上し、TCO が 72%削減されます。ただし、多数の操作を GPU で実行する必要があります。

GPU で実行されている操作を確認するには、Spark イベントログを使用して、GPUColumnarToRow や GPURowToColumnar などの GPU-CPU データ交換操作を検索します。

## Graviton インスタンスの使用

AWS の詳細クラスタ内の Graviton2 インスタンスは、CPU 集中型のジョブに対して最大 26%高速で、コストを 41%節約できます。アグリゲータ、ジョイナ、ランク、ソータートランスフォーメーションを含むシャッフル集約型のジョブでは、違いが見られない可能性があります。

## AMD チップセットの使用

Microsoft Azure の詳細クラスタ内のマスターノードおよびワーカーノード上の AMD チップセット（AMD EPYC 7452 など）は、Intel Xeon よりも 1.2 倍高速です。アグリゲータ、ジョイナ、ランク、ソータートランスフォーメーションや複合式を使用したマッピングを含むシャッフル負荷の高いジョブについては、1.3-1.4 倍高速化することができます。

## 第 7 章

# システムパフォーマンスの最適化

ソース、ターゲット、マッピング、およびマッピングタスクをチューニングした後に、システムのボトルネックを防止してパフォーマンスを最適化するためにシステムのチューニングを行うことを検討してください。

データ統合は、システムリソースを使用して、トランスフォーメーションの処理、タスクの実行、およびデータの読み取りと書き込みを行います。また、データ統合は、システムメモリを使用して、アグリゲータ、ジョイナ、ルックアップ、ソーター、およびランクなどのトランスフォーメーションに対してキャッシュファイルを作成します。

## システムのボトルネックの特定

システムツールを使用すると、Windows システムおよび Linux システムを監視できます。

### Windows でのシステムのボトルネックの特定

タスクマネージャの【パフォーマンスとプロセス】タブで、システム情報を確認できます。タスクマネージャの【パフォーマンス】タブには、CPU の使用率とメモリの総使用量が表示されます。

パフォーマンスモニタを使用すると、より詳細な情報を確認できます。次の表に、Windows のパフォーマンスモニタでグラフを作成するために使用できるシステム情報を示します。

| プロパティ                | 説明   |
|----------------------|--|
| プロセッサ時間 (%)          | 複数の CPU がある場合は、各 CPU のプロセッサ時間の割合を監視します。                        |
| ページ/秒                | ページ/秒が 5 を超えている場合、メモリに過大な負担がかかっている可能性があります (この状態をスラッシングと呼びます)。 |
| 物理ディスク時間 (%)         | 読み取りまたは書き込みの要求を実行したときに、物理ディスクがビジーになる時間の割合を示します。                |
| 物理ディスクキュー長           | 同一のディスクデバイスにアクセスするために待機しているユーザー数を示します。                         |
| 秒あたりのサーバー処理<br>バイト合計 | サーバーとネットワークの間で送信および受信したバイト数を示します。                              |



## Linux でのシステムのボトルネックの特定

Linux には、システムのボトルネックを特定するために使用できるツールがいくつか用意されています。

次のようなコマンドを使用できます。

- top。システム全体のパフォーマンスを表示します。このツールには、システム、およびシステム上で実行されている個々のプロセスに関する CPU 使用率、メモリ使用量およびスワップ使用量が表示されます。
- iostat。データベースサーバーに接続されているすべてのディスクに対するロード操作を監視します。iostat には、ディスクが物理的にアクティブになっている時間の割合が表示されます。ディスクアレイを使用している場合は、iostat ではなくディスクアレイのユーティリティを使用します。
- vmstat。ディスクのスワップ動作を監視します。
- sar。CPU 使用率、メモリ使用量、およびディスク使用量に関する詳細システムアクティビティレポートを表示します。このツールを使用して、CPU のロードを監視できます。ユーザ、システム、アイドル時間および待ち時間の使用率を知ることができます。このツールは、ディスクのスワップ動作の監視にも使用できます。

## システムのボトルネックの除去

システムのボトルネックを特定した後に、システムをチューニングしてボトルネックを除去し、パフォーマンスを向上させます。

システムのボトルネックを除去するには、次のタスクを実行します。

- CPU 使用率が 80%を超過した場合は、同時実行タスクの数をチェックしてください。負荷を変更するか、Secure Agent グループを使用して別のエージェントマシンにタスクを分散することを検討します。負荷が低下しない場合は、プロセッサの追加を検討します。
- スワップが発生する場合は、物理メモリを増設するか、ディスク上のメモリ集約型アプリケーションの数を減らします。
- メモリが極端に圧迫（スラッシング）される場合は、物理メモリの追加を検討します。
- 処理時間の割合が高い場合は、PowerCenter のキャッシュをチューニングして、ディスクに書き込む代わりにインメモリキャッシュを使用するようにします。キャッシュをチューニングしても要求がキューに残っており、ディスクのビジー状態の割合が 50%以上ある場合は、より高速なディスクデバイスにアップグレードします。物理ディスクのキューの長さが 2 を超える場合は、ディスクのアップグレードを検討します。
- I/O の待機時間の割合が高い場合は、より高速なディスクにアップグレードするか、使用された領域が少ない他のディスクを使用することを検討します。例えば、ソースデータ、ターゲットデータ、ルックアップ、ランク、および集計キャッシュファイルがすべて 1 つのディスクに格納されている場合は、これらの一部を別のディスクに移すことを検討します。

# 索引

## C

Cloud アプリケーション統合コミュニティ  
URL [6](#)  
Cloud 開発者コミュニティ  
URL [6](#)

## I

INFA\_MEMORY プロパティ [12](#)  
Informatica Intelligent Cloud Services  
Web サイト [6](#)  
Informatica グローバルカスタマサポート  
連絡先情報 [7](#)

## J

JVMOption プロパティ [12](#)

## M

maxDTMProcesses プロパティ [11](#)

## W

Web サイト [6](#)

## あ

アップグレード通知 [7](#)

## し

システムステータス [7](#)

## す

ステータス  
Informatica Intelligent Cloud Services [7](#)

## は

パフォーマンスのチューニング  
maxDTMProcesses プロパティ [11](#)  
ヒープとメモリの設定 [12](#)

## め

メンテナンスの停止 [7](#)

## ろ

ログレベル [34](#)