ORACLE

Oracle Real Application Clusters (RAC) Exadata 上のキャッシュ・フュージョンのパフォーマンス最適化

2024 年 6 月, Version 23ai

ORACLE

Disclaimer

本文書には、ソフトウェアや印刷物など、いかなる形式のものも含め、オラクルの独占的な所有物である占有情報が含まれます。この機密文書へのアクセスと使用は、締結および遵守に同意した Oracle Software License and Service Agreement の諸条件に従うものとします。本文書と本文書に含まれる情報は、オラクルの事前の書面による同意なしに、公開、複製、再作成、またはオラクルの外部に配布することはできません。本文書は、ライセンス契約の一部ではありません。また、オラクル、オラクルの子会社または関連会社との契約に組み込むことはできません。

本書は情報提供のみを目的としており、記載した製品機能の実装およびアップグレードの計画を支援することのみを意図しています。マテリアルやコード、機能の提供をコミットメント（確約）するものではなく、購買を決定する際の判断材料になさらないでください。本書に記載されている機能の開発、リリース、時期、および価格設定については、弊社の裁量により決定されます。

製品アーキテクチャの性質により、コードが大幅に不安定化するリスクなしに、本書に記載されているすべての機能を安全に含めることができない場合があります。

ORACLE

Table of contents

エグゼクティブ・サマリー **4**

さまざまなパフォーマンス最適化 **5**

Exafusion 5

ゼロ・コピー・ブロック送信 5

Undo ブロックの RDMA 読み取り 5

インメモリ・コミット・キャッシュ 6

共有データ・ブロックおよび UNDO ヘッダーの RDMA 読み取り 6

Broadcast-on-Commit Over RDMA 8

Optimized Object Checkpoints 9

結論 **9**

参考資料 **9**

List of figures

Figure 1. RDMA ベースのキャッシュ・フュージョン プロトコル 7

Figure 2. Oracle Database 21c の SCN メッセージ量の削減 8

ORACLE

エグゼクティブ・サマリー

一般的に Oracle RAC と呼ばれる Oracle Real Application Clusters は、リニアなスケーラビリティと高可用性を実現する Oracle Database の非常に人気の高い機能です。Oracle RAC キャッシュ・フュージョンは Oracle RAC のコンポーネントであり、複数の Oracle RAC インスタンス間のキャッシュを同期することでアプリケーションに変更を加えずに全ての Oracle RAC インスタンスのリソースをシームレスに利用できるようにする役割を担います。キャッシュ・フュージョンでは、キャッシュの同期に専用のプライベート・ネットワークが使用されます。そのため、アプリケーションのスケーラビリティは、基盤となるプライベート・ネットワークの通信速度と帯域幅に依存します。

RDMA over Converged Ethernet（RoCE）といった高度なネットワーク・コンポーネントを採用した Exadata を使用すると、パフォーマンスとスケーラビリティをさらに向上させることができます。オラクルは基盤となるネットワークの速度向上から恩恵を受けるとともに、Exadata で利用できる高度なプロトコルと RDMA 機能を活用できるよう、Oracle RAC キャッシュ・フュージョン・レイヤーの大部分を再設計しました。このホワイト・ペーパーでは、Exadata 上に展開された Oracle データベースで利用可能なこれらの最適化について説明します。

さまざまなパフォーマンス最適化

Exafusion

従来、Oracle RAC のメッセージングは、ネットワーク・ソケットを利用した一般的なネットワーク・モデルを使用して実装されていました。このモデルでは、すべての通信（送受信）が OS カーネルを経由するため、Oracle RAC インスタンス間でメッセージが交換されるたびに、ユーザー空間と OS カーネル空間との間でコンテキスト・スイッチとメモリのコピーが必要となります。Exafusion は RoCE 構成と InfiniBand 構成の Exadata において Oracle Database 12c 以降で提供される次世代ネットワーク・プロトコルであり、**OS カーネル空間を完全にバイパス**してユーザー空間から **direct-to-wire プロトコルでの通信**を実現します。コンテキスト・スイッチと OS カーネルのオーバーヘッドを排除することで、往復のメッセージを 30 µs（マイクロ秒）未満で処理できます。これは、**RAC を汎用サーバー上で動作させる従来型のソケットベースの実装よりも 5 倍高速**です。加えて、Exafusion ではメッセージの送受信に関連する CPU コストも低減されるため、キャッシュ・フュージョンのバックグラウンド・プロセス（LMS プロセス）あたりのブロック転送スループットを向上させ、より多くの処理ができるようになります。**メッセージングの高速化は、通常のアプリケーション・パフォーマンスにとって有益なだけではありません。**ロックの動的なリダイレクト（DRM）、インスタンスや PDB メンバーシップの変更に伴う Oracle RAC の再構成、さらにはインスタンス・リカバリなど、Oracle RAC のあらゆる処理も高速化されます。

Exafusion の採用は、Zero Copy Block Sends や RDMA の導入など、後述する Exadata 上の RAC のパフォーマンス最適化における基盤となります。

Exafusion および本書で説明する最適化では、追加の OS リソースを必要としません。

ゼロ・コピー・ブロック送信

RoCE と InfiniBand ネットワーク・アダプタでは、ゼロ・コピー・メッセージングがサポートされます。OS カーネルが最初にユーザー空間にあるバッファのコピーを作成し、そのコピーを通信層に渡す従来型のメッセージング・プロトコルとは異なり、ユーザー空間にあるバッファは HCA に登録され、HCA はユーザー空間にあるバッファの中身を直接通信層に渡すことができます。バッファをコピーするために必要な CPU サイクルを排除することで、Exadata 上の RAC のキャッシュ・フュージョンの転送速度がさらに最適化されます。

Undo ブロックの RDMA 読み取り

トランザクションのロールバックなどが発生した際には、他の Oracle RAC インスタンスから UNDO ブロックを取得する必要があります。Exadata 上の RAC での Undo ブロック送信は、従来のメッセージベースのプロトコルに代わって RDMA ベースの転送プロトコルを使用するように最適化されています。**RDMA を活用することで、フォアグラウンド・プロセスは他のインスタンスの SGA から UNDO ブロックを直接読み取ることが可能になります。**UNDO ブロックの読取りで他のインスタンスのプロセスが呼び出されることがなくなるため、非 Exadata 環境の RAC で見られるサーバー側処理（CPU）とコンテキスト・スイッチのオーバーヘッドが排除されます。さらに、転送時間は他のインスタンスにおける OS プロセスの輻輳やシステム全体の CPU 負荷の影響を受けなくなるため、クラスタ内の他のインスタンスで**負荷が急増して安定性の問題が発生しても、高速かつ安定した読み取り速度が確保**されます。**RDMA** 読取りは通常 **10 µs 未満**で完了し、メッセージベースのプロトコルで得られる最良の転送時間と比較して **3 倍の改善**が見られます。

インメモリ・コミット・キャッシュ

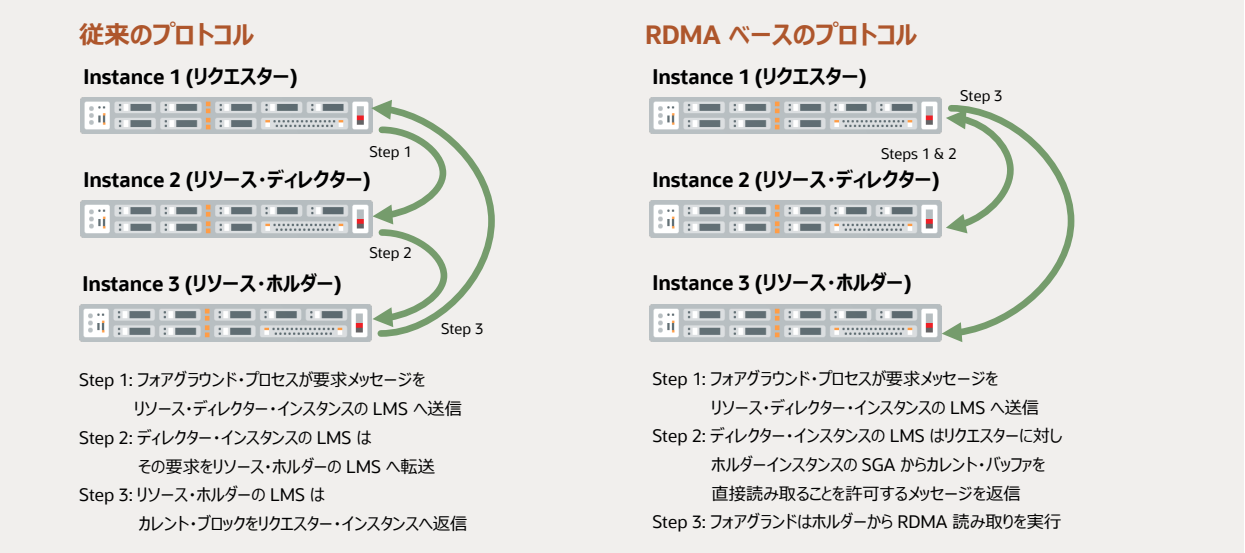
長時間実行されるバッチ・ジョブと同時に実行される問合せがあるアプリケーションでは、“UNDO ヘッダー”の CR ブロックが大量に転送される場合があります。この問題に対処するため、Exadata ではインメモリ・コミット・キャッシュがサポートされています。この機能により、各インスタンスはローカル・トランザクションとその状態（コミット済みか否か）を SGA 内にキャッシュで保持し、お互いにリモートから参照できます。これは、サイズがそれぞれ 8 KB の UNDO ヘッダー・ブロックをリモート・インスタンスから転送するよりも高速です。さらに、複数のトランザクション ID（XID）の状態を単一のメッセージで検索できるため、Oracle RAC におけるラウンドトリップ・メッセージの数をさらに削減し、Commit Cache の参照要求に対応する LMS プロセスの CPU オーバーヘッドも同様に低減されます。インメモリ・コミット・キャッシュを使用すると、この最適化以前は 30 個の XID 検索の為に 30 個の 8 KB ブロック転送が必要であったのに対して、**単一のラウンドトリップ・メッセージで 30 個の XID 検索を一括処理できます。**

コミット・キャッシュの最適化により、“UNDO ヘッダー”送信に対応する多数の“gc cr block 2-way”待機イベントが、より少数の“gc transaction table 2-way”待機イベント（この待機イベントは、Oracle 23ai より前のリリースでは“gc transaction table”と呼ばれていました）に置き換えられる可能性があります。一度の“gc transaction table 2-way”待機イベントは、1 回のラウンドトリップで複数の XID を参照していることを意味します。

共有データ・ブロックおよび UNDO ヘッダーの RDMA 読み取り

Oracle Database 21c では、**キャッシュ・フュージョンにおける RDMA サポートが拡張され、データブロック、領域管理ブロック、および Undo ヘッダーブロックの読み取りにも対応**するようになりました。この最適化は、Undo ブロックの RDMA 読み取りと同様に、他のインスタンスにキャッシュされているデータの読み取りをさらに高速化し、RDMA 経由で読み取る際には LMS プロセスが動作しないために LMS プロセスの CPU 使用量をさらに減少させます。従来は、フォアグラウンド・プロセスがブロック読み取り要求をディレクタ・インスタンスに送信し、ディレクタ・インスタンスがその要求をホルダー・インスタンスに転送するという、3-way のキャッシュ・フュージョン Transfer（”gc current block 3-way”）によって要求が実行されるという流れでした。これは 3 ノード以上の大規模クラスタ上で実行される読み取り集約型の OLTP ワークロードにおいて見られる一般的なアクセス・パターンです。大規模クラスタでは、各インスタンスのサイズが比較的小さくなるため、データがローカル・インスタンスにキャッシュされているよりも別のインスタンスにキャッシュされている可能性の方が高くなります。データと領域管理ブロックで RDMA を使用すると、ディレクタ・インスタンスはリクエスター・インスタンスに対してロック付与（データ読み取りの許可）と共に、要求されたブロックを保持しているホルダー・インスタンスの情報を返します。この情報を使って、リクエスター・インスタンス上のフォアグラウンド・プロセスは、ホルダー・インスタンスから直接 RDMA によってブロックを読み取ることが可能になります。これにより、ディレクタとホルダー・インスタンス間のメッセージ送信が不要となり、読み取り時間の高速化、さらにはホスター・インスタンスの LMS プロセスの（従来はブロックをリクエスターへ転送する必要があった）CPU 使用量が削減されます。

Figure 1. RDMA ベースのキャッシュ・フュージョン プロトコル



このケースでは、フォアグラウンド・プロセスでは、従来の“gc current block 3-way”待機イベントの代わりに次の一連の待機イベントが確認できます。

- “*gc current grant 2-way*” 待機イベントに続いて、
- 短い “*gc current block direct read*” 待機イベント

“gc current block direct read” 待機は通常 10µ 秒以下であり、両待機イベントの合計待機時間は従来の 3-way での転送時間よりも短くなります。

もしリクエスター自身がディレクター・インスタンスであれば、メッセージング無しにデータを読み取る許可を自分自身に付与できるために、前述の”gc current grant 2-way” 待機イベントは不要となります。この場合、一回の”gc current block direct read” 待機イベントで素早く要求を処理されます。この最適化によって、2 ノード・クラスタを含む Oracle RAC で従来発生していた”gc current block 2-way" 待機の一部が置き換えられます。

さらに、もしローカルではないディレクター・インスタンスがホルダー・インスタンスでもあった場合、LMS は許可メッセージを返信して、リクエスターはホルダー（ディレクターでもある）からデータを RDMA で読み取ります。これはディレクターとホルダー・インスタンスが同一である点を除けば、上述した 3-way のシナリオと同じであり、従来の”gc current block 2-way”待機イベントは、”gc current grant 2-way”待機イベントと”gc current block direct read”待機イベントに置き換えられます。この場合は、読み取り時間はそれほど改善されませんが、**LMS がロックを許可するコストはデータブロックを返信するよりも軽いため、RDMA の最適化は LMS プロセスの CPU 使用量を削減するのに役立ちます。**

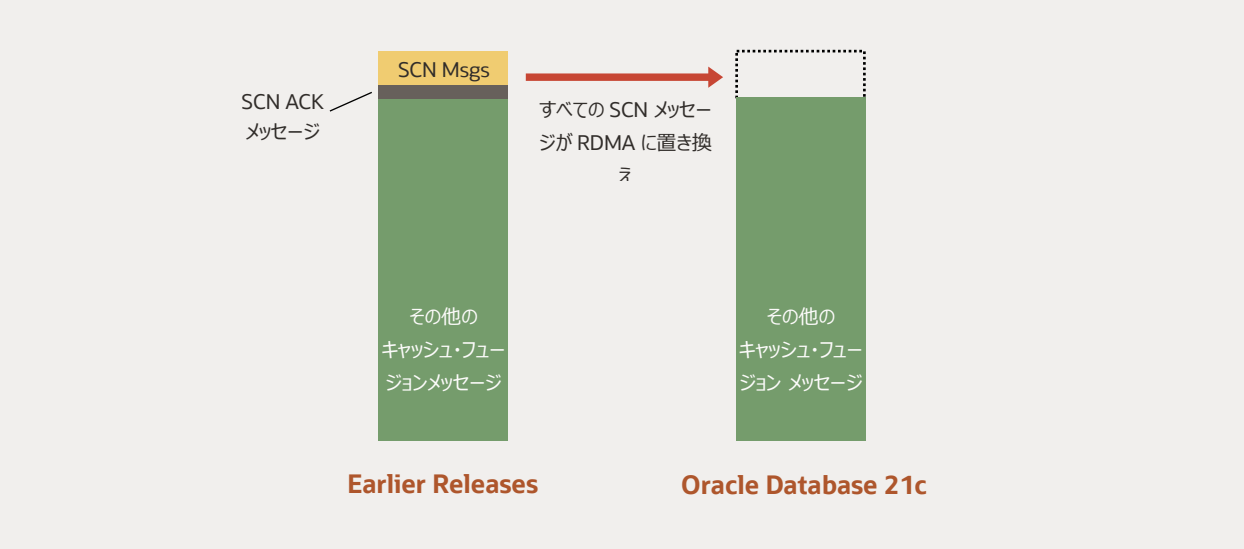
Broadcast-on-Commit Over RDMA

トランザクションをコミットする前に Broadcast-on-Commit プロトコルは、クラスタ内のすべてのインスタンスのシステム変更番号（SCN）が少なくともコミット SCN と同じ高さであることを保証します。これは、Oracle のトランザクションの読み取り一貫性を保証するために必要です。LGWR プロセスはすべてのインスタンスの LMS プロセスに SCN を含むメッセージを送信し、それを受信した LMS プロセスは自身が稼働するインスタンスの SCN を更新して、始めのインスタンス上の LMS プロセスに対して ACK メッセージを返信します。Redo I/O が完了すると、LGWR は redo SCN がすべてのインスタンスによって確認応答されたか否かを確認します。確認応答されていた場合、LGWR はトランザクションを待機しているフォアグラウンド・プロセスに対してコミット操作が完了したことを通知します。もし Redo I/O が完了するまでに redo SCN の確認応答がされなかった場合は、全ての ACK が戻ってくるまでコミットは完了しません。この場合、フォアグラウンドには長い時間の“log file sync”待機イベントが表示されます。

Oracle Database 21c では、Broadcast-on-Commit が RDMA を使用するように最適化されています。RDMA を活用することでメッセージングよりも低遅延が実現され、LMS プロセスの負荷も軽減されます。特に OLTP アプリケーションでは、SCN メッセージがメッセージング通信の一定割合を占めていることが観測されており、大規模クラスタではこの傾向が顕著です。これらのメッセージがレイテンシーのクリティカル・パスになることはほとんどありませんが（基本的には実 I/O 時間の方が長いため）、これらを削減することで LMS プロセスの負荷が減少して余裕が生まれ、システムの急激な負荷上昇に耐えられるようになります。

たとえば、大規模な CRM（OLTP 系）ワークロードを 3 つのインスタンスで構成されるクラスタ上で動かした場合、RAC メッセージ全体の 12%が SCN メッセージであることが分かります。RMDA を使用することで、これらのメッセージは LMS プロセスを呼び出すことがなくなります。

Figure 2. Oracle Database 21c の SCN メッセージ量の削減



Broadcast-on-Commit over RDMA モードでは、LGWR プロセスがアトミックな RDMA 操作を使用してクラスタ内の各インスタンスの SCN を直接更新します。これにより、LMS プロセスのコンテキスト・スイッチに要する時間や他のインスタンスの CPU 負荷状況の影響を受けずに、コミット・プロトコルが高速化されます。

Optimized Object Checkpoints

Exadata スマート・スキャンやパラレル・クエリ（PQ）スキャンが頻繁に発生するワークロードでは、オブジェクトのチェックポイントに関連するパフォーマンスボトルネックが発生する可能性があります。代表的な症状としては、”enq: KO - fast object checkpoint” や “reliable message”の待機時間が長くなることが挙げられます。特に、非常に短いスキャンが高い多重度で実行されるアプリケーションでは、チェックポイントの要求頻度が非常に高くなるため、この傾向が顕著です。

Oracle Database 23ai では、オブジェクトのチェックポイントが必要か否かをクラスタ全体に対して素早く確認するために、高速な RDMA ベースのチェック機能が追加されました。対象オブジェクトに対してグローバル・キャッシュ上にダーティバッファが存在しなければ、チェックポイント処理自体を完全にスキップできます。このチェックは “gc obj ckpt direct read”という待機イベントのもとで実行され、**通常 10 µs 秒未満で完了し、クラスタ内のいかなるバック・グラウンドプロセスも呼び出しません。**この最適化により、AI Vector Search のスキャンは非常に短時間である傾向があるため、Oracle Database 23ai におけるベクター・データベースに対する Exadata Smart Scan のパフォーマンスが大幅に向上します。社内での検証により、この機能が AI Vector Search ワークロードにおけるオブジェクトのチェックポイントの 99% を排除できる ことが確認されています。

結論

本書では、Oracle RAC が Exadata を活用して Oracle RAC のキャッシュ・フュージョン のパフォーマンスをさらに最適化し、アプリケーションの変更を一切必要とせずに大幅なスケーラビリティ向上を実現しているいくつかの例を紹介しました。Exadata を通じて、オラクル は最新のハードウェア機能を最大限に活用するようデータベースソフトウェアを設計することで顧客の生産性を大きく高めており、さらなるイノベーションに今後も投資し続けます。

参考資料

- Oracle Real Application Clusters (RAC) White Paper
- Oracle RAC Internals – The Cache Fusion Edition
- Oracle RAC features on Exadata

ORACLE

Connect with us

+1.800.ORACLE1 までご連絡いただくか、**oracle.com**をご覧ください。
北米以外の地域では、**oracle.com/contact** で最寄りの営業所をご確認いただけます。

blogs.oracle.com　facebook.com/oracle　twitter.com/oracle

著者: Atsushi Morimura, Namrata Jampani, Anil Nair　共著者: Neil Macnaughton, Avneesh Pant, Michael Zoll