ORACLE

Oracle ExadataでのOracle Real Application Clusters（RAC）キャッシュ・フュージョン・パフォーマンスの最適化

2021年5月

Public

免責事項

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、および時期については、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。

目次

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、線形の水平スケーラビリティと高可用性を実現するOracle Databaseのオプションです。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、Oracle RACインスタンス間のキャッシュを同期することで、変更を加えずにアプリケーションがすべてのOracle RACインスタンスの演算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・ネットワークの待機時間と帯域幅に依存します。

RDMA over Converged Ethernet（RoCE）やInfiniBand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。たとえば、Exadataでは、Oracle RACインスタンスはバッファをケーブルに直接送信し、オペレーティング・システム（OS）カーネルを迂回します。その結果、待機時間が極めて短いブロック送信が実現し、CPUコストが大幅に低減されます。ExadataでのOracle RACは、パフォーマンスが極めて重要なトランザクション・コミットで待機時間を排除する新しいプロトコルも使用しています。このホワイト・ペーパーでは、Oracle 12c以来実装されているExadata固有のこのような最適化について説明します。

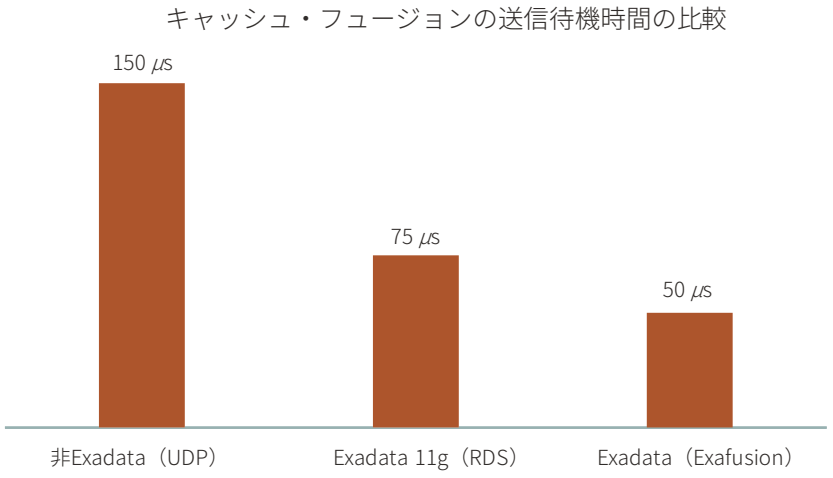
さまざまなパフォーマンス最適化

Exafusion

これまでは、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メッセージが交換されるたびに、ユーザーの領域とOSカーネル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、12c以降、（RoCEとInfiniBandの双方の）Exadataで提供されている次世代ネットワーク・プロトコルであり、**OSカーネルを完全に迂回して**、ユーザーの領域から**直接ケーブルにメッセージを送信**することが可能です。Exafusionでは、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、50 µs（マイクロ秒）未満でラウンドトリップ・メッセージを処理できます。**従来型のソケットベースの実装よりも3倍高速であり**、メッセージングにRDSプロトコルを使用した**Exadataの第一世代と比較して、さらに33％向上しています。**

加えて、メッセージの送受信に関連するCPUコストもExafusionでは低減されるため、ブロック送信スループットを向上させ、LMSプロセスが飽和状態になる前にヘッドルームを増加させることができます。

メッセージングの高速化は、実行時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。動的ロック・リマスタリング（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、インスタンス・リカバリなど、**Oracle RACのあらゆる操作も高速化されます。**



Exafusionの採用は、ゼロ・コピー送信やRDMAの採用など、後述するExadataでのOracle RACのパフォーマンス最適化における基盤となります。

Exafusionと本書で後述する最適化では、追加のOSリソースを稼働する必要はありません。Exafusionを有効化すると、“ps”で、IPC0バックグラウンド・プロセスのRSSメモリ使用量が多いことに気づくかもしれません。これは、インスタンスで実行されているすべてのプロセスに代わって、Oracleインスタンスがすべてのバッファをホスト・チャネル・アダプタ（HCA）に登録（固定）しているためであり、**過剰なメモリ使用やメモリ・リークを示しているわけではありません。**

詳細については、MOS Note 2407743.1を参照してください。

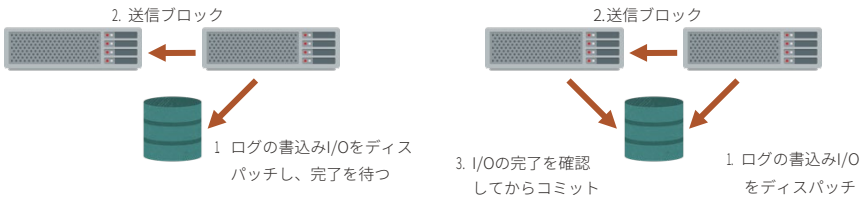
ゼロ・コピー・ブロック送信

RoCEおよび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カーネルがまずユーザー領域バッファのコピーを作成してからそのコピーをケーブルに配置する従来型のメッセージング・プロトコルとは異なり、ユーザー領域バッファはHCAに登録され、HCAはユーザー領域バッファのコンテンツを直接ケーブルに配置します。Oracle RAC 12c以降、Exadataではインスタンス間の通信にこの機能を使用しています。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、**ゼロ・コピー送信のないExafusionと比較して、送信の待機時間が最大で5 %向上しました。**

Smart Fusion Block Transfer

これまでは、Oracle RACインスタンスはREDOログのフラッシュが完了するのを待ってから、他のインスタンスに使用済みブロックを送信する必要がありました。これは、DML操作が頻繁なOLTPシステムでは一般的なアクセス・パターンです。REDOログのフラッシュは、インスタンス障害時にデータベースの整合性を保つために行われます。つまり、頻繁に変更され、保留中のREDOがあるブロックでは、インスタンス間の送信待機時間は、常にREDOログのフラッシュのI/O待機時間に依存し、I/Oパフォーマンスが断続的に低下することで生じる外れ値の対象となっていました。

Oracle RAC 12cでは、Smart Fusion Block Transfer最適化が使用されるため、Exadataストレージ・サーバーに対する**REDO I/Oが処理中**になると、Oracle RACインスタンスがブロックを送信できるようになります。Oracle RAC LMSプロセスでは、I/Oの完了確認を受領する前にブロック送信の開始が許可されるため、要求元インスタンスのセッションは、REDO I/Oがまだ保留されている間に、そのブロックへのアクセスを開始できます。要求元インスタンスは、I/Oの完了を確認してから、同じブロックに対するさらなる変更をコミットします。コミット・プロセスは、I/Oがまだ完了していない場合、“remote log force - commit”待機イベントを待つ必要があります。これは非常にまれな例であり、I/Oの著しい外れ値がある場合にのみ発生します。
このようなI/Oの外れ値は、スマート・フラッシュ・ロギング機能を搭載したExadataではほとんど排除されています。Smart Fusion Block Transfer最適化を使用すると、Oracle RACインスタンス全体で同時実行性を向上させ、アプリケーションの全体的なパフォーマンスを改善できます。この最適化の結果、ホット・ブロックを同時に更新するワークロードで、“*gc current block busy*”**の待機時間が3分の1**に短縮されます。



従来のプロトコル　　Smart Fusion Block Transfer

UNDOブロックのRDMA読取り

トランザクションのロールバックなどがある場合、UNDOブロックを他のOracle RACインスタンスからフェッチする必要があります。Oracle RAC 18cでは、UNDOブロック送信は、従来型のメッセージベースのプロトコルの代わりに、RDMAベースの送信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・プロセスが、リモート・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できます。**UNDOブロックの読取りで、リモート・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、常に従来型のOracle RAC通信の一部であったサーバー側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。さらに送信待機時間も、リモート・インスタンスのOSプロセスやシステムの全体的なCPU負荷による影響を受けなくなるため、リモート・インスタンスで**負荷が急増した場合でも、確実な読取り待機時間が確保されます。**リモート・ブロックのRDMA読取りは、通常は**10 µs未満**で完了し、Exafusionを使用した従来型のメッセージベースのプロトコルで達成できる最短待機時間から**5分の1に短縮**されています。

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・ジョブと同時問合せがあるアプリケーションには、大量の“UNDOヘッダー”CRブロック送信が表示される場合があります。Oracle 18cでは、**Exadataにインメモリ・コミット・キャッシュが追加されました。**各インスタンスによって、ローカル・トランザクションのキャッシュと各キャッシュの状態（コミット済み、または未コミット）がSGAに保持され、キャッシュはリモートで検索できます。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・インスタンスに送信するよりも高速です。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のラウンドトリップ・メッセージ数が削減され、リモート検索リクエストを処理するLMSプロセスのCPUオーバーヘッドも低減されます。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の8 KBブロック送信が必要であったのに対して、**単一のラウンドトリップ・メッセージにおいて、最大で30のXID検索を一括処理できます。**

コミット・キャッシュ最適化を使用すれば、“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*”待機イベントに置き換えられることが予想されます。単一の“*gc transaction table 2-way*”待機イベントは、1つのラウンドトリップに複数のXIDのリモート検索があることを表します。

注：*“gc transaction table 2-way”*待機イベントは、Oracle 21c以降のリリースで使用されます。それ以前のリリース（Oracle 18cおよび19c）では、代わりに*“gc transaction table”*待機イベントが使用されます。

高速索引分割

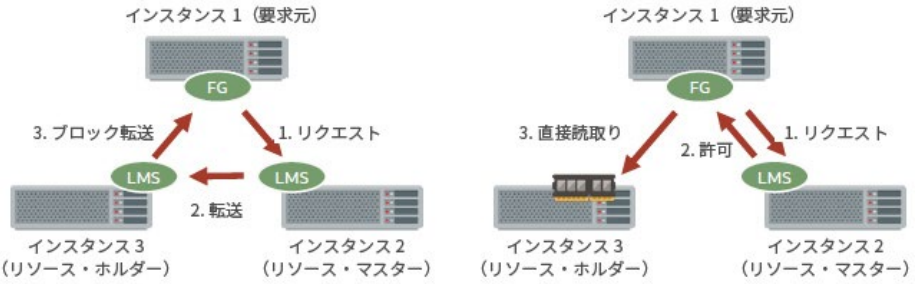
Bツリー索引のリーフ・ブロック分割がある場合（索引が適切に増加するOLTPワークロードでは頻繁に見られます）、あらゆるOracle RACインスタンスのリーフおよびブランチ・ブロックの分割にアクセスするアプリケーションは、分割操作が完了するまで待機する必要があります。これは、アプリケーションのパフォーマンスで断続的な停止（アクティビティがほぼゼロになる期間）を招く場合があります。これまでは、このような待機は、TXエンキュー（“*enq: TX-index contention*”待機イベント）の下で実装されていました。Oracle 19cのExadataでは、このような分割の待機は、グローバル・エンキューの代わりに、より低コストのキャッシュ・フュージョン・ベースの仕組みを使用するように最適化されています。**高速索引分割の待機は、従来のTXエンキュー待機に置き換わる新しい“*gc index operation*”待機イベント（21c以降は *“index split completion”*）の下で管理されます。**

Persistent Memory Commit Accelerator

Exadata X8Mでは、**RDMA書込みのREDOログI/Oをストレージ・サーバーの永続メモリ**に実装するPersistent Memory Commit Acceleratorが導入されています。この最適化により、REDOログのフラッシュI/Oパフォーマンスが大幅に向上し、大量の使用済みバッファを共有するシステムでインスタンス間の同時実行性がさらに改善されます（「Smart Fusion Block Transfer」を参照）。

共有データ・ブロックおよびUNDOヘッダーのRDMA読取り

Oracle 21cでは、**キャッシュ・フュージョンのRDMAサポートが拡張され、データ・ブロック、スペース・ブロック、およびUNDOヘッダー・ブロックの読取りがサポートされるようになりました。**これにより、18cでのUNDOブロックをRDMAで読み取る最適化と同様に、リモート・インスタンスでキャッシュされているデータの読取りがさらに高速化されるほか、データがRDMA経由で読み取られる際にLMSプロセスが起動されないことから、LMSプロセスのCPU使用量がさらに低減されます。従来の方法では、フォアグラウンド・プロセスがマスター・インスタンスにブロック読取りリクエストを送信し、その後マスター・インスタンスが、実際にブロックを所有しているホルダー・インスタンスにリクエストを転送するという、3方向のキャッシュ・フュージョン転送（*"gc current block 3-way"*）によってリクエストが実行されます。これは、ノードが3つ以上の大規模なクラスタ上で実行される、読取りが非常に多いOLTPワークロードでは一般的なアクセス・パターンです。大規模なクラスタでは、たいていの場合、各インスタンスのサイズは小さいため、データがローカル・インスタンスにキャッシュされる可能性は低くなりますが、別のインスタンスにキャッシュされる確率は高くなります。データおよびスペース・ブロックでRDMAを使用すると、マスター・インスタンスは要求されたブロックのホルダー・インスタンスに関する情報とともに、ロック権限（データを読み取る権限）を要求元に返信します。この情報を使用して、要求元クライアントは、ホルダー・インスタンスからブロックを直接RDMAで読み取ることができます。これにより、マスター・インスタンスとホルダー・インスタンス間のメッセージが不要になり、読取り待機時間が短縮されます。さらに、ホルダー・インスタンスのLMSプロセスのCPU使用量が削減されます（従来はブロックを要求元に転送する必要がありました）。



従来のプロトコル　　RDMAベースのプロトコル

この場合、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を確認できます。

- *"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"*待機イベントは**通常10 us以下**であり、許可と読取りの待機イベントを合わせた合計待機時間は、従来の3方向転送の待機時間よりも短くなるのが一般的です。

要求元自身がマスター・インスタンスであれば、メッセージなしでデータを読み取る権限を自分自身に付与できるため、上記の例の *“gc current grant 2-way”*待機イベントは不要になります。この場合、単一の *“gc current block direct read”*待機イベントで素早くリクエストを処理できます。これにより、2ノード・クラスタを含む従来のOracle RACで使用されていた一部の *“gc current block 2-way”*待機イベントが置き換えられます。

さらに、リモートのマスター・インスタンスがホルダー・インスタンスでもある場合、LMSは許可メッセージを返信し、要求元はホルダー（マスターでもある）からデータをRDMAで読み取ります。これはマスター・インスタンスとホルダー・インスタンスが同じであることを除けば、上述した3方向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、読取り待機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コストよりも少ないため、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と同じ大きさであることを保証します。これは、Oracleトランザクションの読取り一貫性（CR）プロパティを保証するために必要です。これまでは、Broadcast-on-Commitプロトコルはメッセージを使用して、SCNをクラスタ内のすべてのインスタンスに一斉送信していました。
LGWRプロセスは、すべてのインスタンスのLMSプロセスにメッセージでSCN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SCN 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っているフォアグラウンド・プロセスに、コミット操作が完了したことを通知します。REDO I/Oが完了するまでにREDO SCNが確認応答されなかった場合は、すべてのSCN ACKが受信されるまでコミットは完了しません。この場合、クライアントには待機時間が長い *“log file sync”*待機イベントが表示されます。

Oracle 21cでは、次の理由により、Broadcast-on-CommitがRDMAを使用するように最適化されています。

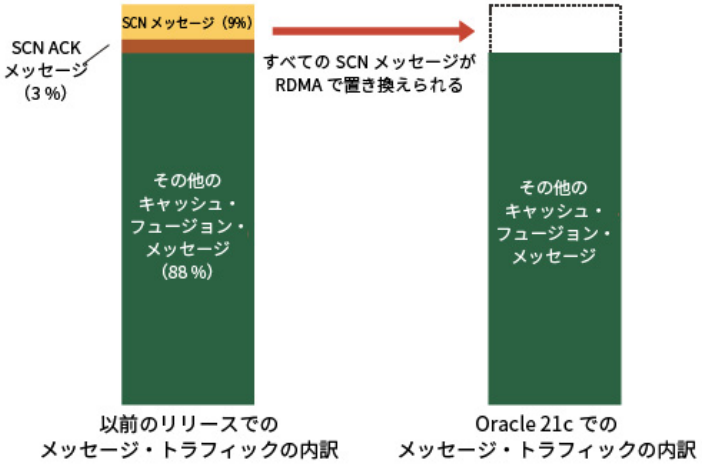
1. RDMAの待機時間がメッセージよりも短いため。

ExadataでI/O待機時間が改善されると、メッセージを使用したSCNのブロードキャストがボトルネックになる可能性があります。

2. LMSプロセスの負荷を軽減するため。

OLTPアプリケーションを実行すると、特にインスタンス数が多いクラスタでは、SCNメッセージがメッセージ・トラフィックの相当な割合を占めていることが分かります。これらのメッセージが待機時間のクリティカル・パスになることはほとんどありませんが（実際のI/O時間の方が通常は長いため）、これらのメッセージを削減すると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、3つのインスタンスで構成されるクラスタ上で大規模なCRM（OLTP）ワークロードを実行したところ、Oracle RACメッセージ全体の12 %がSCNブロードキャストでした。RDMAを使用することで、これらのメッセージでLMSプロセスを呼び出す必要がなくなります。



Broadcast-on-Commit over RDMAモードでは、LGWRプロセスはアトミックなリモート操作を使用してクラスタ内の各リモート・インスタンスのSCNを直接更新します。リモートLMSプロセスのコンテキスト・スイッチに要する時間やリモート・インスタンスのCPU負荷状況の影響を受けないため、コミット・プロトコルが高速化されます。

結論

Oracle RACでExadataのハードウェアの進化を活用して、Oracle RAC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することで、アプリケーションに変更を加えずにアプリケーションのスケーラビリティを大幅に向上させる例をいくつかご紹介しました。オラクルは、市場で入手できる最新のハードウェア・テクノロジーを生かすようソフトウェアを設計することで、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（Oracle RAC）ホワイト・ペーパー
- Oracle RAC Internals – The Cache Fusion Edition
- Oracle RAC 12c Practical Performance Management and Tuning
- Oracle RAC features on Exadata
- Oracle RAC 12c Release 2 – New Availability Features

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