Oracle Maximum
Availability Architecture

同期REDO転送のベスト・プラクティス

Oracle Data GuardとOracle Active Data Guard

Oracleホワイト・ペーパー | 2015年3月

目次

概要 1
Data Guard の同期転送 – 概要 2
同期転送のパフォーマンス 3
同期転送の機能強化 5
Oracle Database 11g Release 2 5
Oracle Database 12c 6
構成のベスト・プラクティス 6
tcpソケット・バッファ・サイズをBDPの3倍に設定する 6
スタンバイREDOログの構成 8
SDUサイズは65535に設定します 9
最適なシステム・パフォーマンスのための十分なリソースの構成 9
Fast Sync（SYNC NOAFFIRM）の使用 9
データ損失ゼロの構成での、Exadataのパフォーマンス向上の検討 9
チューニング 10
同期転送でデータ整合性を維持する方法について 14
パフォーマンスの評価 15
Oracle Database 11.2のSYNCパフォーマンスの評価方法 19
Oracle Database 12cのSYNCパフォーマンスの評価方法 20
ログ・ファイル同期待機の診断 21
Data Guard Fast Sync 24
結論 24

概要

Oracle Maximum Availability Architecture（Oracle MAA）とOracle Data Guardの組み合わせは、ミッション・クリティカルなOracle Databaseのシングル・ポイント障害を排除できるもっとも包括的なソリューションです。このソリューションは、本番データベースの同期された1つ以上の物理レプリカを遠隔地に保持することによって、もっとも単純かつ経済的な方法でデータの損失と停止時間の発生を防ぎます。何らかの理由で本番データベースが使用できなくなると、クライアント接続が、迅速に（また一部の構成では透過的に）、同期されているレプリカにフェイルオーバーされ、すぐにサービスがリストアされます。Active Data GuardはData Guardの基本機能を拡張したもので、レポート作成アプリケーション、非定型の問合せ、データ抽出、および高速増分バックアップを本番データベースの読取り専用コピーにオフロードできるため、コストのかかる無駄な冗長性がなくなります。Active Data Guardはリアルタイムなデータ保護と可用性に完全に特化しており、Oracle Databaseと緊密に統合されているため、ストレージのリモート・ミラー化やその他のホストベースのレプリケーション・ソリューションとは違い、データ保護、パフォーマンス、可用性の低下や複雑さの問題がありません。

Data GuardとActive Data GuardはOracle対応の唯一のレプリケーション・ソリューションであり、実行中の本番データベースの同期済みコピーにデータ損失ゼロで確実にフェイルオーバーできます。Active Data Guard 12cの新機能では、本番データベースからさまざまな距離にある同期レプリカを使用した、データ損失ゼロの保護が拡張されています。本番データベースのパフォーマンスに影響したり、フェイルオーバー操作が複雑になったりすることはありません。このため、データベース、クラスタ、サイト、地域、および地理的に停止が発生しても、Oracleデータベースの高可用性とデータ保護を実現できます。

本書では、Data GuardやActive Data Guardの構成で同期REDO転送を使用するための、Oracle MAAのベスト・プラクティスについて説明します。本書は、最大可用性モードや最大保護モードを使用したData GuardとActive Data Guardの構成について、実用的な知識を持っているデータベース管理者を対象としています。これらのモードでは、同期REDO転送とData Guard管理リカバリ・プロセス（MRP）の統合によって、本番データベースの計画外停止が発生してもデータ損失ゼロが保証されます。

Data Guard の同期転送 – 概要

Data Guardの構成には、プライマリ・データベースと呼ばれる本番データベースと、スタンバイ・データベースと呼ばれる最大30の直接接続レプリカが含まれます。プライマリ・データベースとスタンバイ・データベースは、Oracle Net Servicesを使用したTCP/IPを介して接続されます。データベース同士の通信が可能であれば、設置場所に関する制約はありません。スタンバイ・データベースは、最初はプライマリ・データベースのバックアップから作成されます。Data Guardは、プライマリ・データベースのREDO（トランザクションを保護するためにすべてのOracle Databaseによって使用される情報）を送信し、スタンバイ・データベースに適用して、プライマリ・データベースとすべてのスタンバイ・データベースを自動的に同期します。

Data Guardの転送サービスは、プライマリ・データベースからスタンバイ・データベースへのREDOの送信に関するすべての側面を処理します。ユーザーがプライマリ・データベースでトランザクションをコミットすると、REDOレコードが生成され、ローカルのオンライン・ログ・ファイルに書き込まれます。Data Guardの転送サービスは、プライマリ・データベースのログ・バッファ（システム・グローバル領域内で割り当てられているメモリ）からスタンバイ・データベースへ同じREDOを同時に直接送信し、スタンバイ・データベースはスタンバイREDOログ・ファイルにREDOを書き込みます。Data Guardによる転送は、以下の理由から非常に効率的です。

» Data Guardを使ったメモリからの直接送信により、プライマリ・データベースでのディスクI/Oオーバーヘッドが回避されます。これは、他の多くのホストベース・レプリケーション・ソリューションで、ディスクからのデータ読取りと変更の送信によって、プライマリ・データベースのI/Oが増加するのとは異なります。

» Data Guardでは、データベースREDOだけが送信されます。これは、リアルタイム同期を維持するためにすべてのファイルのすべての変更ブロックを送信しなければならないストレージのリモート・ミラー化とは完全に対照的です。オラクルが実施したテストによると、ストレージのリモート・ミラー化では、Data Guardよりも、最大7倍のネットワーク・データが送信され、27倍のネットワークI/O操作が必要であることが示されています。ストレージのリモート・ミラー化と比較した場合のData Guardの利点について詳しくは、『*Oracle Active Data Guardとストレージのリモート・ミラー化の比較』を参照してください*。[1]

  Data Guardでは、同期と非同期の2つの転送サービスを選択できます。同期REDO転送（本書のテーマ）では、REDOが受信され、ディスク（スタンバイREDOログ・ファイル）に書き込まれたというスタンバイ・データベースからの確認を待ってから、プライマリ・データベースがコミットの成功を伝える信号をアプリケーションに送信します。

同期転送と、Data Guardの適用サービスによるトランザクション・セマンティクスの高度な認識との結合により、プライマリ・データベースに突然障害が発生しても、データ損失ゼロの保護が保証されます。

また、Data Guardには3種類の操作モードがあり、コスト、可用性、パフォーマンス、データ保護のバランスを調整できます。表1を参照してください。各モードでは、プライマリ・データベースとスタンバイ・データベースの接続が失われた場合のData Guard構成の動作を定義します。3つのモードのうち、2つは同期転送を使用します。

[1] http://www.oracle.com/technetwork/jp/database/availability/dataguardvsstoragemirroring-2082185-ja.pdf

表1：Data Guardの保護モード

モード	データ損失のリスク	転送	スタンバイ・データベースからの確認がない場合：
最大保護	データ損失ゼロ 二重障害からの保護	SYNC	トランザクションのREDOがディスクに書き込まれたことを示すスタンバイ・データベースからの確認を受信してから、コミットの成功をアプリケーションに通知します。通知が受信されるまで、本番データベースは処理を進めることができません。
最大可用性	データ損失ゼロ 単一障害からの保護	SYNC FAST SYNC FAR SYNC	スタンバイ・データベースからの確認を受信するか、またはNET_TIMEOUTしきい値に指定された期間が過ぎたら、（いずれか早い方によって）コミットの成功をアプリケーションに通知します。ネットワークやスタンバイ・データベースの停止は、本番データベースの可用性に影響しません。
最大パフォーマンス	最小限のデータ損失の可能性	ASYNC	プライマリはスタンバイからの確認を待つことなく、コミットの成功をアプリケーションに通知します。データ損失ゼロは保証できません。

同期転送のパフォーマンス

またData Guardには、ストレージのリモート・ミラー化に基づく同期ソリューションと比べて、多くのパフォーマンス上の利点があります。前述のとおり、Data GuardではREDOが送信されるだけです。これに対し、ストレージのリモート・ミラー化でData Guardと同じリアルタイム保護を維持するには、すべての変更をすべてのブロックに送信する必要があります。つまり、ストレージのリモート・ミラー化の場合、Data Guardだけでなく、すべての書込み先によって送信されるデータ量（データファイル、オンライン・ログ・ファイル・グループの追加メンバー、アーカイブ・ログ・ファイル、制御ファイル、フラッシュバック・ログ・ファイルなど）が送信されます。この影響は重大です。たとえば、データファイルへの書込みを行うOracleプロセスはDatabase Writer（DBWR）と呼ばれ、DBWRの速度低下の要因が、データベース・パフォーマンスに大きく影響する可能性があります。ストレージの同期的リモート・ミラー化は、仕様上、DBWRに影響を与えます。ミラー化されたボリューム間のラウンドトリップ・ネットワーク待機時間（RTT）によって、DBWRによる書込みのたびに遅延が発生するためです。Data Guardは、プライマリ・データベースのDBWRに影響を与えないように設計されています。

オラクルは、本番データベースでの同期転送の影響を特定するために、徹底的にテストを行いました。その代表的な2つのテスト結果は、次のとおりです。このデータは、パフォーマンスへの影響に関する一般的な概要を示したものです。実際の本番ワークロードへの影響の予測に使用しないでください。

アプリケーションによって、同期レプリケーションへの耐性は異なります。アプリケーションの同時実行性、セッション数、トランザクション・サイズ（バイト単位）、セッションのコミット頻度、およびログ・スイッチの頻度が異なるため、ラウンドトリップ・ネットワーク待機時間（RTT）、帯域幅、およびログ・ファイルの書込みI/Oパフォーマンスがすべて同じでも、影響はアプリケーションによって異なります。一般的に、ラウンドトリップ・ネットワーク待機時間が5ミリ秒超の場合より、5ミリ秒未満の場合の方が同期転送に適しています。実際のワークロードでの同期レプリケーションの影響について特定の結論を出す前に、必ずテストを行ってください。

テスト1：OLTPワークロード、スモールinsert

Swingbench（パブリック・ドメインの負荷生成ツール[2]）を使用して、30MB/秒でREDOを生成したOLTPワークロードをシミュレートしました。その結果、パフォーマンスへの影響は1ミリ秒のRTTで3%、5ミリ秒のRTTで5%でした（図1を参照）。

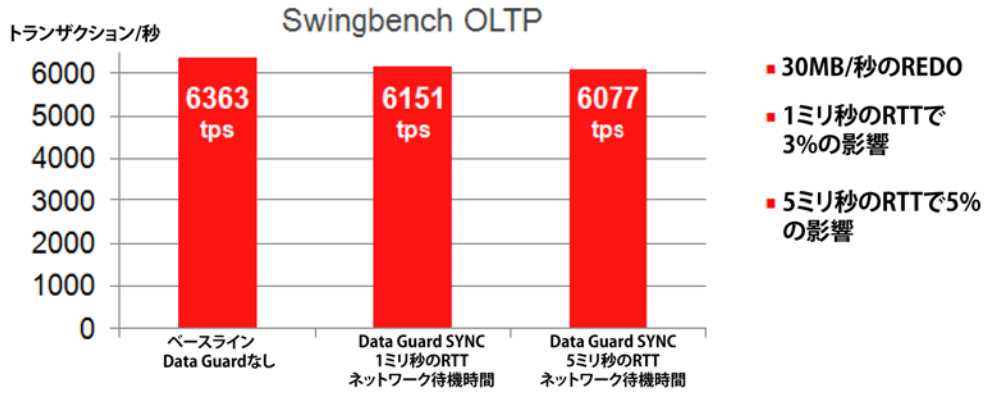

図1：OLTPワークロードの同期転送のパフォーマンスへの影響 - スモールinsert

テスト1のSwingbenchワークロードには、次の特性があります。

» ランダムDML、1ミリ秒のシンク・タイム、400ユーザー、1秒あたり6000以上のトランザクション、30MB/秒のピークREDO速度

» スモールinsert：トランザクションあたり5KのREDOサイズ、120の論理読取り、30のブロック変更

» 3回のテスト実行：Data Guardなし、および2回のData Guardテスト実行（1ミリ秒と5ミリ秒のラウンドトリップ・ネットワーク待機時間）

» Oracle Exadataの2ノードOracle Real Application Clusters（Oracle RAC）データベース、Oracle 11.2.0.4、Exadata Smart Flash Logging、およびライトバック・フラッシュ・キャッシュ

テスト2：OLTPワークロード、ラージinsert

同じシステムとデータベース構成を使用した2回目のテストでは、平均トランザクション・サイズが440Kに増加し、それに対応してREDOスループットが増えた場合のOLTPワークロードへの影響をプロファイルしました。パフォーマンスへの影響は、1ミリ秒未満のRTTで1%、2ミリ秒のRTTで7%、5ミリ秒のRTTで8%でした（図2を参照）。

[2] http://dominicgiles.com/swingbench.html

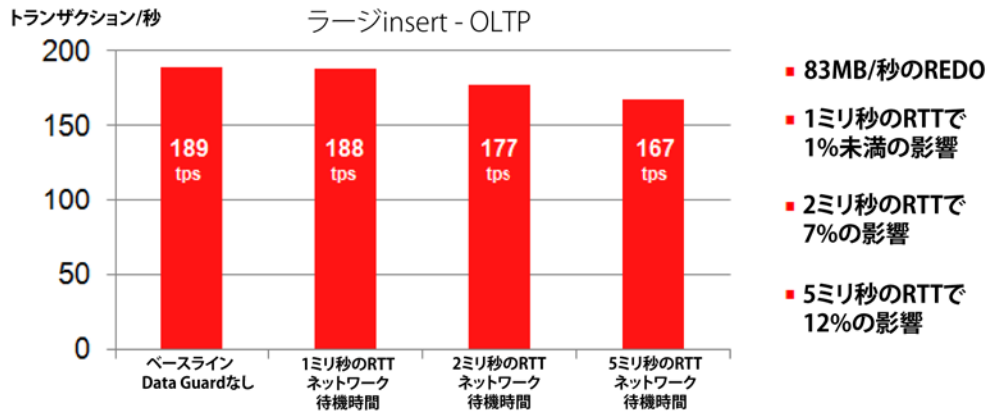

図2：OLTPワークロードの同期転送のパフォーマンスへの影響 - ラージinsert

テスト2のSwingbenchワークロードには、次の特性があります。

» ラージinsertのOLTPワークロード：1秒当たり180以上のトランザクション、83MB/秒のピークREDO速度、ランダム表

» トランザクション・プロファイル440KのREDOサイズ、6000の論理読取り、トランザクションあたり2100のブロック変更

» Data Guardなしのベースラインの後、1ミリ秒、2ミリ秒、5ミリ秒のRTTネットワーク待機時間でData Guardを使用

同期転送の機能強化

同期転送は、多くのOracle Databaseリリースにわたって進化してきました。同期転送の技術的詳細と、関連するOracle MAAのベスト・プラクティスは、Data GuardおよびActive Data Guardと同じです。本書のData Guardのベスト・プラクティスは、すべてActive Data Guardにも適用できます。

Oracle Database 11g Release 2

Data Guard 11g Release 2では、プライマリ・データベースのローカル・オンライン・ログ・ファイルに対するLGWRのREDO書込みと並行して、リモード・スタンバイにREDOが送信され、同期レプリケーションに必要な合計ラウンドトリップ時間が短縮されるため、同期転送のパフォーマンス上の影響が少なくなります。これは、以前のData Guardリリースから改善された点です。以前のData Guardでは転送の際、リモート・スタンバイへのREDOの送信前に、ローカル・ログ・ファイルへの書込みが完了するまで待機していました。合計ラウンドトリップ時間が短縮されるため、データ損失ゼロの同期構成で、プライマリ・データベースとスタンバイ・データベースを地理的にさらに離れた場所に配置できるようになりました。または、待機時間の短いネットワークを使用することで、同期レプリケーションによるプライマリ・データベースのパフォーマンスへの影響をほぼゼロにまで軽減できます。この同じアーキテクチャが、Oracle Database 12cで使用されています。

Oracle Database 12c

Oracle Database 12cのData Guard Fast Sync（SYNC NOAFFIRM）を使用すると、データ損失ゼロの同期構成のパフォーマンスを簡単に上げることができます。Fast Syncを使用すると、スタンバイはスタンバイREDOログ・ファイルへのディスクI/Oを待たなくても、REDOをメモリ内に受信したらすぐにプライマリ・データベースを確認できます。これにより、プライマリとスタンバイの間の合計ラウンドトリップ時間が大幅に短縮されるため、プライマリ・データベースのパフォーマンスへの同期トランスポートの影響が軽減されます。Fast Syncでは、スタンバイ・データベースのI/Oが完了する前にプライマリ・データベースとスタンバイ・データベースの両方で障害が同時発生するとデータが損失する危険性がごくわずかながら存在します。ただし、その危険性のある時間は非常に短く（もう一方の障害が数ミリ秒の間に発生する必要がある）、そのような状況は極めて特異なものであるため、発生する可能性はほとんどありません。Fast Syncは、Data Guard 12cに含まれています（Active Data Guardのライセンスは不要です）。Fast Syncは、管理者が明示的に有効にする必要があります。この操作を実行しないと、Data Guard 12cの同期REDO転送のデフォルトの動作が、Data Guard 11gと同じ（SYNC AFFIRM）になります。

Active Data Guard 12cには、Far Syncという新しい機能が含まれます。Active Data GuardのFar Syncを使用すると、プライマリ・データベースとスタンバイ・データベースが数百～数千マイル離れていても、データ損失ゼロの保護が可能です。またこの際、広域ネットワーク全体で、本番データベースへの同期通信の影響を軽減または解消できます。Far SyncとOracle Advanced Compressionを組み合わせると、オフ・ホストでのREDO転送圧縮が可能になり、本番システムの帯域幅のオーバーヘッドをゼロにしておくことができます。本書では、Far Syncについては説明しません。Far Syncの詳細とベスト・プラクティスについては、『*Oracle Active Data Guard Far Sync - Zero Data Loss at Any Distance*』[3]を参照してください。

構成のベスト・プラクティス

次のMAAベスト・プラクティスは、Data Guard同期REDO転送の構成のパフォーマンス上の影響を最小限に抑えて、本番データベースをデータ損失ゼロで保護できるように設計されています。

tcpソケット・バッファ・サイズをBDPの3倍に設定する

最適なネットワーク・スループットを実現するための、TCPの送受信ソケット・バッファ・サイズの最小推奨設定値は、プライマリ・システムとスタンバイ・システムの間のネットワーク・リンクの帯域幅遅延積（BDP）と同じです。また、BDPより高い値を設定すると、徐々に改善される場合があります。たとえば、待機時間が長い高帯域幅ネットワークをシミュレートするMAAテストの場合、TCPの送受信ソケット・バッファ設定をBDPの3倍まで増やすと、スループットが少しずつ段階的、継続的に増加しました。

BDPは、ネットワークの帯域幅と待機時間の積です。ソケット・バッファ・サイズは、Oracle Netパラメータ RECV_BUF_SIZEおよびSEND_BUF_SIZEを使用して、ソケット・バッファ・サイズ設定がOracle TCP接続のみに影響するように設定されます。オペレーティング・システムのソケット・バッファ・サイズに制限がある場合があります。このような場合、Oracleでより大きい値を使用するには調整が必要です。たとえばLinuxの場合、パラメータnet.core.rmem_maxとnet.core.wmem_maxでソケット・バッファ・サイズが制限されており、RECV_BUF_SIZEとSEND_BUF_SIZEより大きい値に設定する必要があります。

[3] http://www.oracle.com/technetwork/database/availability/farsync-2267608.pdf

たとえば、帯域幅が622Mbで待機時間が30ミリ秒の場合、パラメータRECV_BUF_SIZEとSEND_BUF_SIZEの最小サイズは次のように計算します。

帯域幅遅延積（BDP） = 帯域幅 x 待機時間

BDP = 622,000,000（帯域幅）/8 x 0.030（待機時間） = 2,332,500バイト

この例を前提とすると、送受信ソケット・バッファの最適なサイズは次のように計算されます。ソケット・バッファ・サイズ = 3 x BDP

= 2,332,500（BDP） x 3

= 6,997,500バイト

ソケット・バッファのサイズは、オペレーティング・システム・レベルかOracle Netレベルで設定できます。ソケット・バッファ・サイズの要件は、（ネットワーク条件によって）かなり大きくなる場合があります。このサイズをOracle Netレベルで設定して、telnetなどの通常のTCPセッションで、より多くのメモリが使用されないようにすることを推奨します。オペレーティング・システムによっては、すべての送受信ソケット・バッファの最大サイズを設定するパラメータが含まれる場合があるので注意してください。Oracle Netでより大きなソケット・バッファ・サイズを使用するには、必ずこれらの値を調整する必要があります。

Oracle Netの場合、sqlnet.oraで次のパラメータを使用して、すべての接続の送受信ソケット・バッファ・サイズをグローバルに設定できます。

```
RECV_BUF_SIZE=6997500

SEND_BUF_SIZE=6997500
```

単にData Guard転送と関連する接続のバッファ・サイズを増やしたい場合は、転送用のOracle Netエイリアスとスタンバイ・ホストのリスナーで、RECV_BUF_SIZEとSEND_BUF_SIZEを構成することを推奨します。次の例では、送受信ソケット・バッファ・サイズが、特定の接続記述子の説明属性として設定されています。

```
standby =
  (DESCRIPTION=
  (SEND_BUF_SIZE=6997500)
  (RECV_BUF_SIZE=6997500)
       (ADDRESS=(PROTOCOL=tcp)
       (HOST=stby_host)(PORT=1521))
    (CONNECT_DATA=
       (SERVICE_NAME=standby)))
```

ソケット・バッファ・サイズは、Data Guard構成内のすべてのデータベースで同一に構成する必要があります。スタンバイ側や受信側では、sqlnet.oraファイルかlistener.oraファイルでこの構成を行うことができます。listener.oraファイルでは、特定のプロトコル・アドレスまたは記述のバッファ領域パラメータを指定できます。

```
LISTENER =
  (DESCRIPTION=
       (ADDRESS=(PROTOCOL=tcp)
       (HOST=stby_host) (PORT=1521)
       (SEND_BUF_SIZE=9375000)
       (RECV_BUF_SIZE=9375000)))
```

スタンバイREDOログの構成

オンラインREDOログとスタンバイREDOログでは、REDOログ・サイズを4GB、またはピークREDO速度/分 x 20分以下にする必要があります。ピークREDO速度を抽出するには、ピーク・ワークロード期間中のAWRレポート（バッチ処理、四半期末や年末の処理など）を参照してください。平均ではなく、ピーク・ワークロードを使用することは非常に重要です（平均を使用すると、ピークREDO速度が分かりにくくなり、ネットワーク帯域幅のプロビジョニングが不十分になる可能性があるためです）。表2は、REDO速度と最小REDOログの推奨サイズの簡単なマッピングを示しています。

表2：REDOログの推奨サイズ

EMレポートまたはAWRレポートに従ったピークREDO速度	REDOログ・グループの推奨サイズ
5MB/秒以下	4GB
25 MB/秒以下	16 GB
50 MB/秒以下	32GB
50MB/秒未満	64 GB

オンラインREDOログのサイズを適切に設定したら、同じサイズのスタンバイREDOログを作成する必要があります。**これは、単一メンバーしか含まれないスタンバイREDOログ・グループのパフォーマンスで重要です。**また、各REDOログ・スレッド（スレッドはOracle RACデータベース・インスタンスに関連付けられています）では、スタンバイREDOログの数 ＝REDOログ・グループの数 ＋1です。

スタンバイREDOログを追加すると、スタンバイREDOログでスタンバイ・データベースが待機する可能性がなくなります。たとえば、プライマリ・データベースに2つのインスタンス（スレッド）が含まれており、各スレッドに3つのオンライン・ログ・グループが含まれる場合、プライマリ・データベースと各スタンバイ・データベースで、8つのスタンバイREDOログを事前構成する必要があります。さらに、プライマリ・データベースやスタンバイ・データベースが対称型のReal Application Clustersでない場合（8ノードのプライマリOracle RACクラスタと2ノードのスタンバイOracle RACクラスタなど）、（構成内の最大クラスタに基づいて）プライマリ・データベースとスタンバイ・データベースのスタンバイREDOログの数を同じにして、すべてのスレッドを表す必要があります。スタンバイREDOログのサイズと、プライマリのオンラインREDOログのサイズは、必ず完全に同じである必要があります。

単一メンバーのスタンバイREDOログ・グループは、必ず使用可能な最速のディスクグループに置く必要があります。これは、スタンバイ・ログ・ファイルの書込み時間を、プライマリ・データベースの最適なパフォーマンスのログ・ファイルI/Oと同等にするためです。

SDUサイズは65535に設定します

Oracle Net Servicesの場合、セッション・データ・ユニット（SDU）のOracle Net設定のサイズを調整することで、データ送信を制御できます。オラクルのテスト結果によると、SDUを最大値の65535に設定すると、SYNC転送のパフォーマンスを上げることができます。SDUは、ローカル・ネーミング構成ファイル（TNSNAMES.ORA）とリスナー構成ファイル（LISTENER.ORA）のSDUパラメータを使用して接続ごとに設定するか、SQLNET.ORAファイルのプロファイル・パラメータDEFAULT_SDU_SIZEを使用して、すべてのOracle Net接続用に設定できます。

ASYNC転送では新しいストリーミング・プロトコルが使用されており、SDUサイズをデフォルトより増やしても非同期構成のパフォーマンスは向上しないので、注意してください。

最適なシステム・パフォーマンスのための十分なリソースの構成

十分なリソースを確保することは、特にプライマリ・データベースとスタンバイ・データベースのログ・ファイルI/Oや、プライマリ・データベースとスタンバイ・データベースの間のネットワーク帯域幅の場合、同期Data Guard構成のパフォーマンスにとって重要です。Oracle Database 12cのFast SYNCを使用した場合、スタンバイ・データベースのI/Oが低速でも本番データベースのパフォーマンスに影響しませんが、最適な結果を出すには、プライマリ・データベース・ログ・ファイルの書込み用のI/Oパフォーマンスと、REDOボリュームの送信用のネットワーク帯域幅を十分に確保することも必要です。

Fast Sync（SYNC NOAFFIRM）の使用

Fast Sync（SYNC NOAFFIRM）を使用すると、リモートRFSの書込みを待機しているすべてのセッションの処理を、書込みの完了後ではなく書込みの送信後に進めることができます。この結果、全体的なSYNCリモート書込み待機イベントが減るため、最適なパフォーマンスを実現できます。

データ損失ゼロの構成での、Exadataのパフォーマンス向上の検討

Exadataには、SYNC構成のデプロイに最適なシステムを実現するための、いくつかの機能が含まれています。

» Smart Flash Logging：Exadata Smart Flash Cacheには、ログ書込みI/Oの待機時間を短縮するための、Exadata Smart Flash Loggingと呼ばれる特別なアルゴリズムが実装されています。ユーザー・トランザクションのコミット時間や、重要な更新の実行時間は、ログ書込みの待機時間に大きく影響を受けます。Smart Flash Loggingでは、Exadataストレージのフラッシュ・キャッシュとExadataディスク・コントローラの高速RAMメモリを組み合わせて利用することで、ログ書込みの待機時間を大幅に短縮し、（フラッシュ・ベースのストレージ・ソリューションを含む）すべてのストレージ・ソリューションで頻繁に発生する待機時間のスパイクを回避します。Exadata Smart Flash Loggingアルゴリズムは Exadata固有のもので、プライマリとスタンバイのログ書込みが高速化されます。

» ネットワーク：Exadataコンピュート・ノードには、1GigEと10GigEの複数のネットワーク・アダプタが搭載されています。これらのネットワーク・アダプタは高可用性のために事前構成されており、高速転送を処理するための広帯域幅を備えています。

» ストレージ・パフォーマンス：Exadataは、業界標準のスケールアウト・データベース・サーバー、インテリジェントなスケールアウト・ストレージ・サーバー、最先端のPCIフラッシュ・ストレージ・サー

バー、およびすべてのサーバーとストレージを接続する超高速内部InfiniBandファブリックを搭載した最新アーキテクチャです。Exadata独自のソフトウェア・アルゴリズムには、ストレージのデータベース・インテリジェンス、PCIベースのフラッシュ、およびInfiniBandネットワークが実装されており、他のプラットフォームより高いパフォーマンスと大容量をより低コストで実現できます。

チューニング

Data Guardのパフォーマンスは、プライマリ・システムとスタンバイ・システム、これらのシステムの接続ネットワーク、およびIOサブシステムのパフォーマンスに直接依存します。Data Guard構成のトポロジと、そのData Guardパフォーマンスとの関係を理解すれば、Data Guardアーキテクチャに起因すると誤解されがちなインフラストラクチャの弱点を解消できます。

構成の前提条件

Data Guardアーキテクチャは非常に合理的かつ効率的ですが、他のアプリケーションと同様に、十分なパフォーマンスを発揮するために必要な合理的な前提条件があります。

プライマリ・データベース

» フォアグラウンドを効率的にポストするための、LGWRの十分なCPU使用率

» ピーク速度時にローカル・ログ書込みで短いI/O待機時間を維持するための、十分なI/O帯域幅

» ピークREDO速度ボリュームと、同じインタフェースの他のネットワーク・アクティビティを組み合わせて処理できるネットワーク・インタフェース

» チューニングやトラブルシューティング用に収集されるAWR、ASH、OSwatcherのプライマリ・データ

ネットワーク：

» プライマリ・データベースとスタンバイ・データベースの間のラウンドトリップ・ネットワーク待機時間（RTT）は、プライマリ・データベースがパフォーマンスSLAに違反するほど大きくしないようにする必要があります。つまり、プライマリ・データベースとスタンバイ・データベースの間の距離には、実質的に制限があるということです。ラウンドトリップの待機時間は、距離が伸びれば長くなるためです。サポート可能な最長待機時間や、パフォーマンスへの影響の推定に使用できる方程式の有無に関する問い合わせはよく受けますが、ネットワーク待機時間への対応能力は、アプリケーションによって異なります。アプリケーションによって同時実行性、セッション数、トランザクション・サイズ（バイト単位）、セッションのコミット頻度、およびログ・スイッチの頻度が異なるため、ラウンドトリップ・ネットワーク待機時間、帯域幅、およびログ・ファイルの書込みI/Oパフォーマンスがすべて同じでも、影響はアプリケーションによって異なる場合があります。一般的に、ラウンドトリップ・ネットワーク待機時間が5ミリ秒超の場合より、5ミリ秒未満の場合の方が同期REDO転送に適しています。

» ピークREDO速度（安定状態とギャップの解決時）と、同じネットワークを共有する他のネットワーク・アクティビティの組み合わせをサポートするための、十分なネットワーク帯域幅。Point-to-Pointのネットワーク帯域幅は、ネットワークの最小帯域幅によるネットワークのセグメント、スイッチ、ルーター、インタフェースによって抑制されるので注意してください。ネットワーク・ルートの90%が10GigEであり、既存のスイッチやネットワーク・インタフェースが1GigEのみをサポートしている場合、ネットワークの最大帯域幅は1GigEです。

» netstatやネットワーク監視のstatを収集する必要があります

注：ネットワークで発生する最大の問題は、一貫性のないネットワーク応答と、ネットワーク帯域幅の不足です。

スタンバイ・データベース

- スタンバイREDOログに効率的に書き込むための、RFS（スタンバイ・データベースでREDOを受信するData Guardプロセス）の十分なCPU使用率。
- ピーク速度時にローカル・ログ書込みで短いI/O待機時間を維持するための、十分なI/O帯域幅
- ピークREDO速度ボリュームと、同じインタフェースの他のネットワーク・アクティビティを組み合わせて受信できるネットワーク・インタフェース
- スタンバイのstatspackデータ、ASHデータ、およびOSwatcherデータを収集する必要があります

注：スタンバイ・データベースで発生する最大の問題は、I/O帯域幅の不足によって、スタンバイ・ログ書込みの待機時間が長くなることです。この問題は、Oracle 12cのData Guard Fast Syncや、Exadataを使用することで軽減できます。

システム・リソースの監視

プライマリ・ホストとスタンバイ・ホストのシステム・リソースの監視方法に関する情報を以下に示します。

CPUの監視

uptime、mpstat、sar、dstat、およびtopの各ユーティリティを使用して、CPU使用率を監視できます。システムのCPUコアがプロセス作業の実行用にすべて使用されている場合、他のプロセスは、CPUコアやスレッドが解放されるか、スケジューラによってCPUがそのプロセス・コードを実行するように切り替えられるまで、待機する必要があります。キューのプロセスが多すぎる状態が頻繁に発生すると、システム・パフォーマンスのボトルネックとなる可能性があります。

コマンドmpstat -P ALLとsar -u -P ALLを実行すると、各CPUコアと、すべてのCPUコアを平均した、CPUの使用統計情報が表示されます。

%idleの値は、CPUでシステム・コードやプロセス・コードが実行されなかった時間の割合を示しています。%idleの値が0%に近い場合、すべてのCPUコアのほとんどの時間帯で、システムが実行するワークロードのためにCPUが限界になっています。特に%idleが0%に近い場合、システム・コード（%systemまたは%sys）の実行にかかる時間の割合は通常、30%を超えないようにする必要があります。

システム・ロードの平均は、一定の期間内で、CPUコアで実行されているプロセス、実行を待機しているプロセス、またはディスクI/Oアクティビティが完了するまで待機しているプロセスの平均数を表しています。ビジー状態のシステムでは、uptimeやsar -qで報告されるロード平均が、CPUコア数の2倍を超えないようにする必要があります。ロード平均が長期間、CPUコア数の4倍を超えると、システムがオーバーロードになります。

ロード平均（ldavg-*）に加えて、sar -qコマンドを実行すると、現在実行を待機しているプロセス（run-queueサイズ、runq-sz）の数と、プロセスの合計数（plist_sz）が報告されます。runq-szの値は、CPUの飽和も示します。

ユーザーやアプリケーションによるシステムの使用時に応答性の問題が発生しないような、通常ロードでのシステムの平均ロードを決定し、このベンチマークからの経時的な逸脱を調べます。ロード平均の大幅な増加は、パフォーマンス上の重大な問題を示している可能性があります。

メモリ使用率の監視

sar -rコマンドを実行すると、%memused（使用中の物理メモリの割合）を含む、メモリ使用率の統計情報が報告されます。

» sar -Bを実行すると、pgscank/s（kswapd daemonによってスキャンされる1秒あたりのメモリ・ページ数）やpgscand/s（直接スキャンされる1秒あたりのメモリ・ページ数）を含む、メモリ・ページングの統計情報が報告されます。

» sar -Wを実行すると、pswpin/sとpswpout/s（スワップイン/スワップアウトされる1秒あたりのページ数）を含む、スワッピングの統計情報が報告されます。

%memusedが100%近くで、スキャン速度が継続的に200ページ/秒である場合、システムがメモリ不足になります。

システムの実メモリや物理メモリが不足して、スワップ領域が使用されるようになると、パフォーマンスが大幅に低下します。スワップ領域が不足すると、プログラムやオペレーティング・システム全体がクラッシュする場合があります。freeやtopで、使用可能なスワップ領域がほとんど残っていないことが表示される場合は、メモリも不足しているということです。

dmesgコマンドからの出力には、起動時に検出された、物理メモリの問題の通知が含まれる可能性があります。

I/Oの監視：

iostatコマンドを実行すると、平均データ転送速度に対してデバイスがアクティブである時間を監視することで、ブロックI/Oデバイスのローディングを監視できます。この情報を使用してシステム構成を調整し、ディスクとホスト・アダプタ全体でI/Oロードのバランスを取ります。

iostat -xを実行すると、ブロックI/Oアクティビティに関するさらに詳しい統計情報が、1秒間隔で報告されます。この情報には、%util（デバイスへのI/Oリクエストの処理にかかったCPU時間の割合）や、avgqu-sz（そのデバイスに発行されたI/Oリクエストの平均キュー長）が含まれます。%utilが100%近くであるか、avgqu-szが1より大きい場合は、デバイスの飽和が発生しているため、ディスクやストレージを追加して、ストレージのI/O帯域幅を増やす必要があります。

また、sar -dコマンドを使用して、ブロックI/Oアクティビティ（%utilとavgqu-szの値を含む）について報告することもできます。

iotopユーティリティを使用すると、過剰なディスクI/Oの原因となっているプロセスを特定できます。iotopのユーザー・インタフェースはtopと似ています。iotopの上部には、ディスクの入出力使用の合計がバイト/秒単位で表示されます。iotopの下部には、各プロセスのI/O情報（ディスクの入出力使用（バイト/秒単位）、ディスクからのページのスワップインやI/Oの待機にかかる時間の割合、コマンド名など）が表示されます。左右の矢印キーを使用してソート・フィールドを変更し、[A]キーを押して、I/Oユニットのバイト/秒と合計バイト数を切り替えます。または[O]キーを押して、すべてのプロセスの表示と、I/Oを実行しているプロセスのみの表示を切り替えます。

ネットワークの監視：

ip -s linkコマンドを実行すると、ネットワークの統計情報とすべてのネットワーク・デバイスのエラー（送信バイト（TX）と受信バイト（RX）の数を含む）が表示されます。droppedフィールドとoverrunフィールドには、ネットワーク・インタフェースの飽和のインジケータが表示されます。

ss -sコマンドを実行すると、各プロトコルのサマリー統計情報が表示されます。

インタフェース経由の現在の送信速度を監視するには、sar –n DEVコマンドを使用します。

データベースの待機イベントの評価

システム・リソースやネットワーク・リソースでボトルネックが発生していないことを確認したら、データベース待機イベントを評価できます。この操作は、プライマリ・データベースではAWRレポートを使用して実行できますが、スタンバイ・データベースではスタンバイStatspackレポートを使用します（スタンバイStatspackについて詳しくは、My Oracle Support Note 454848.1を参照してください）。Oracle Database 11.2のData Guard固有の待機イベントについては、表3を参照してください。

表3：Data Guard 11.2に関連する待機イベント

イベント名	説明
ARCH wait on ATTACH	すべてのアーカイバ・プロセスで、新しいRFS接続の起動にかかる時間を監視します。
ARCH wait on SENDREQ	すべてのアーカイバ・プロセスで、アーカイブ・ログのローカル・ディスクへの書込みと、リモート書込みにかかる時間を監視します。
ARCH wait on DETACH	すべてのアーカイバ・プロセスで、RFS接続の削除にかかる時間を監視します。
LNS wait on ATTACH	すべてのLNSプロセスで、新しいRFS接続の起動にかかる時間を監視します。
LNS wait on SENDREQ	すべてのLNSプロセスで、受信したREDOのディスクへの書込みと、リモート・アーカイブREDOログの開閉にかかる時間を監視します。
LNS wait on DETACH	すべてのLNSプロセスで、RFS接続の削除にかかる時間を監視します。
LGWR wait on LNS	ログ・ライター（LGWR）プロセスで、LNSプロセスからのメッセージ受信の待機にかかる時間を監視します。
LNS wait on LGWR	LNSプロセスで、ログ・ライター（LGWR）プロセスからのメッセージ受信の待機にかかる時間を監視します。
LGWR-LNS wait on channel	ログ・ライター（LGWR）プロセスまたはLNSプロセスで、メッセージ受信の待機にかかる時間を監視します。

Oracle 11.2の待機イベントは、12.1.0.1以降のすべてのデータベース・バージョンで、次の表4のイベントに置き換えられています。

表4：Oracle Database 12c以降のData Guardに関連する待機イベント

イベント名	説明
ARCH Remote Write	ARCnバックグラウンド・プロセスが、ネットワーク書込み操作が完了するまでブロックされて待機する時間（センチ秒単位）。
ASYNC Remote Write	非同期ストリーミングのRFSWRITE操作の時間（センチ秒単位）。リープの停止時間やストリーミング・ネットワークの送信時間が含まれます。この時間は、TTnn（REDO転送スレーブ）のバックグラウンド・プロセスによって蓄積されます。
Redo Transport Attach	すべてのネットワーク・プロセスで、Connect、Logon、RFSATTACHの実行にかかる時間（センチ秒単位）。
Redo Transport Close	ARCn、NSSn、TTnnのプロセスで、RFSCLOSEとRFSRGSTRの操作にかかる時間（センチ秒単位）。
Redo Transport Detach	すべてのネットワーク・プロセスで、RFSDETACHとDisconnectの実行にかかる時間（センチ秒単位）。
Redo Transport Open	ARCn、NSSn、TTnnのバックグラウンド・プロセスで、RFSCREATとRFSANNCEの操作にかかる時間（センチ秒単位）。
Redo Transport Ping	ARCnのバックグラウンド・プロセスで、RFSPING操作にかかる時間（センチ秒単位）。
Redo Transport Slave Shutdown	LGWRでNSSnプロセスとTTnnプロセスのシャットダウンと終了にかかる時間（センチ秒単位）。
Redo Transport Slave Startup	LGWRでNSSnプロセスとTTnnプロセスの起動と初期化にかかる時間（センチ秒単位）。
Redo Writer Remote Sync Complete	LGWRで、完了したネットワーク書込みをリモートの宛先にリープするためにかかる時間（センチ秒単位）。
Redo Writer Remote Sync Notify	LGWRで、ネットワーク書込みをリモートの宛先に発行するためにかかる時間（センチ秒単位）。
Remote SYNC Ping	LGWRとNSSnのバックグラウンド・プロセスで、同期PING操作の実行にかかる時間（センチ秒単位）。
SYNC Remote Write	LGWRでSYNC RFSWRITE操作の実行にかかる時間。

同期転送でデータ整合性を維持する方法について

次のアルゴリズムによって、Data Guard同期構成でのデータ整合性が維持されます。

» プライマリ・データベースでのLGWR REDO書込みと、Data GuardのNSSネットワークREDO書込みは同じです

» Data Guardのフェイルオーバー操作中（プライマリが使用不可の場合）を除き、プライマリのオンラインREDOログにREDOが書き込まれないと、スタンバイ・データベースでData Guard管理リカバリ・プロセス（MRP）を使用してREDOを適用することはできません。NSSとLGWRでは、REDOを同期的に送信するほか、スタンバイREDOログからスタンバイ・リカバリを適用できる安全なREDOブロック境界に関する情報もやり取りします。このため、スタンバイで受信済みの可能性があるが、プライマリのオンラインREDOログに対してコミットが通知されていないREDOが適用されることはありません。

可能性のある障害シナリオとしては、次のようなものがあります。

» プライマリ・データベースのLGWRでオンラインREDOログに書き込むことができないと、LGWRとインスタンスがクラッシュします。インスタンスやクラッシュ・リカバリは、オンラインREDOログの最後にコミットされたトランザクションにリカバリされます。コミットされていないトランザクションはロールバックされます。現在のログは完了してアーカイブされます。

» スタンバイでは、対応するオンラインREDOログと一致する、正しいサイズ値を使用して、一部のスタンバイREDOログを完了しています。いずれかのREDOブロックがSRLから失われた場合は、（REDOログ全体を再送信せずに）これらのブロックを送信します。

» プライマリ・データベースがクラッシュして、データ損失ゼロのフェイルオーバーが自動または手動で実行される場合、Data Guardフェイルオーバー操作の一部で"ターミナル・リカバリ"が実行され、現在のスタンバイREDOログの読取りとリカバリが行われます。SRLのすべてのREDOを適用してリカバリが完了すると、新しいプライマリ・データベースが使用可能になり、新たに完了したログ・グループがアーカイブされます。新規/既存のすべてのスタンバイ・データベースでは、オンラインREDOログのREDOがすべて破棄され、一貫性のあるSCNにフラッシュバックされて、新しいプライマリ・データベースからのアーカイブのみが適用されます。Data Guard環境が、（新しい）プライマリ・データベースと再度同期されます。

パフォーマンスの評価

SYNC環境のパフォーマンスを評価する場合は、さまざまな待機イベントの相関関係を把握することが重要です。SYNCの有効化による影響は、アプリケーションによって異なります。その理由を理解するには、コミットの発行時にLGWRで実行される作業に関する次の説明を参照してください。

1) フォアグラウンド・プロセスによって、コミットするLGWRがポストされます（"ログ・ファイル同期"の開始）。同時コミット・リクエストがキューにある場合は、未処理のコミット・リクエストがLGWRによってすべてバッチ処理されるため、REDOストランドが連続することになります。
2) LGWRは、CPUが使用できるまで待機します
3) LGWRでREDO書込みが開始されます（"REDO書込み時間"の開始）
4) Oracle RACの場合、LGWRから他のインスタンスに対して、現在の書込みがブロードキャストされます
5) 前処理の後、SYNCスタンバイがある場合は、LGWRによってリモート書込みが開始されます（“SYNCリモート書込み”の開始）
6) LGWRでローカル書込みが発行されます（"ログ・ファイル・パラレル書込み"）
7) SYNCスタンバイがある場合、LGWRはリモート書込みが完了するまで待機します
8) I/Oステータスの確認後に、LGWRで"REDO書込み時間/SYNCリモート書込み"が終了します
9) Oracle RACの場合、LGWRはbroadcast ackが使用できるまで待機します
10) LGWRによって、オンディスクSCNが更新されます
11) LGWRによって、フォアグラウンドがポストされます
12) フォアグラウンドは、CPUが使用できるまで待機します
13) フォアグラウンドで、"ログ・ファイル同期"が終了します

パフォーマンスを評価するには、次の方法を推奨します。

バッチ・ロードでもっとも重要な要素は、経過時間を監視することです。これらのプロセスのほとんどを、決められた期間内に完了する必要があるためです。これらの操作のデータベース・ワークロードは、通常のOLTPワークロードと大幅に異なります。たとえば、書込みサイズがはるかに大きくなる場合があります。このため、ログ・ファイル同期の平均を使用すると、正確な判断や比較ができません。

OLTPワークロードの場合は、（AWRの）1秒あたりのトランザクション・ボリュームと、AWRレポートのREDO速度（1秒あたりのREDOサイズ）を監視することを推奨します。これで、アプリケーション・スループットと、SYNCの有効化によるアプリケーション・スループットへの影響を明確に把握できます。

SYNCの有効化による影響を評価しない方法：

SYNCの有効化による影響を評価する場合、通常はプライマリのログ・ファイル同期待機イベントを最初に確認します。SYNCを有効化する前にログ・ファイル同期待機の平均が3ミリ秒であり、有効化した後に6ミリ秒であった場合、SYNCによるパフォーマンスへの影響は100%と推定されます。ログ・ファイル同期待機時間を使用してSYNCの影響を推定することは推奨しません。平均は非常にあてにならず、SYNCによる応答時間やスループットへの実際の影響は、イベントが示す内容よりはるかに小さい場合があるためです。

ユーザー・セッションがコミットされると、LGWRではCPUを確保し、I/Oを送信し、I/Oの完了まで待機してから、再度CPUを確保して、コミットが完了したフォアグラウンド・プロセスをポストします。このすべての期間が、'ログ・ファイル同期'待機イベントとなります。LGWRの作業中にはほとんどの場合、コミット処理の前に、LGWRの作業終了まで待機する必要があるコミット中の他のセッションがあります。待機中のセッションのサイズと数は、アプリケーションに含まれるセッション数と、これらのセッションのコミット頻度によって決まります。通常、このようなコミットのバッチをアプリケーションの同時実行性と呼びます。

たとえば通常、ログ書込み（ログ・ファイル・パラレル書込み）に0.5ミリ秒、コミットの処理（ログ・ファイル同期）に1ミリ秒かかり、コミットごとに平均で100セッションを処理するとします。ストレージ層に異常があり、1コミットのログ書込みI/Oの完了に20ミリ秒かかった場合、ログ・ファイル・パラレル書込みのための長時間の待機セッションは1つだけですが、ログ・ファイル同期の待機セッションは最大2,000にもなる可能性があります。1つの長時間の外れ値に対して、多数のセッションが待機していると、ログ・ファイル同期の平均が非常に偏ってしまう場合があります。

特定の期間内の、ログ・ファイル同期待機イベントのv$event_histogramからの出力については、表5を参照してください。

表5：ログ・ファイル同期待機イベントのv$event_histogramからの出力

ミリ秒	待機の数	待機の合計に対する割合
1	17610	21.83%
2	43670	54.14%
4	8394	10.41%
8	4072	5.05%
16	4344	5.39%
32	2109	2.61%
64	460	0.57%
128	6	0.01%

この出力結果によると、ログ・ファイル同期待機時間の92%が8ミリ秒未満であり、ほとんど（86%）が4ミリ秒未満です。8ミリ秒超の待機（外れ値）の割合は全体の8%だけですが、（コミットのバッチによる）この外れ値に関するセッション数が原因で、平均が偏っています。このように偏りが出るため、ログ・ファイル同期の平均待機時間をSYNCの影響の評価メトリックとして使用することは適切ではありません。

外れ値の原因は何でしょうか。

同期転送では、プライマリやスタンバイでのI/Oの中断や、ネットワーク待機時間のスパイクも、ログ・ファイル同期の大きな外れ値の原因となる場合があります。この問題は通常、スタンバイ・システムのI/Oサブシステムが、プライマリのI/Oサブシステムより劣っている場合に発生します。スタンバイ・システムに（開発データベースやテスト・データベースなど）複数のデータベースをホストすることで、IOの応答に悪影響を与える可能性があることは、よくあります。前述のように、iostatを使用してI/Oを監視して、ディスクが最大IOPSに近づいているかどうかを特定することが重要です。これがSYNC書込みのパフォーマンスに影響するためです。

おそらく、外れ値の最大の原因の1つは、頻繁なログ・スイッチです。プライマリでログ・スイッチが発生した場合に、スタンバイで何が起こるかについて考えてみます。

1. スタンバイのRFSでは、スタンバイREDOログ・ヘッダーへの更新を終了する必要があります
2. 次にRFSでは、追加のヘッダー更新によって、新しいスタンバイREDOログにスイッチされます
3. ログ・スイッチによって、スタンバイでのフル・チェックポイントが強制されます。このため、バッファ・キャッシュ内のすべての使用済みバッファがディスクに書き込まれ、書込みI/Oのスパイクの原因になります。このため、スタンバイ・ストレージ・サブシステムとプライマリ・データベースのパフォーマンスが異なる非対称構成の場合は、IO待機時間が長くなります。
4. 以前のスタンバイREDOログは、読取りと書込みのI/Oが増加するため、アーカイブする必要があります。

図3のグラフは、約30MB/秒という大きな挿入ワークロードを使用して作成しています。このグラフには、ログ・スイッチが含まれていた期間の各リモート書込みの時間がミリ秒単位で表示されています。また、その増加が各ログ・スイッチで発生するリモート書込み時間であることを示しています。

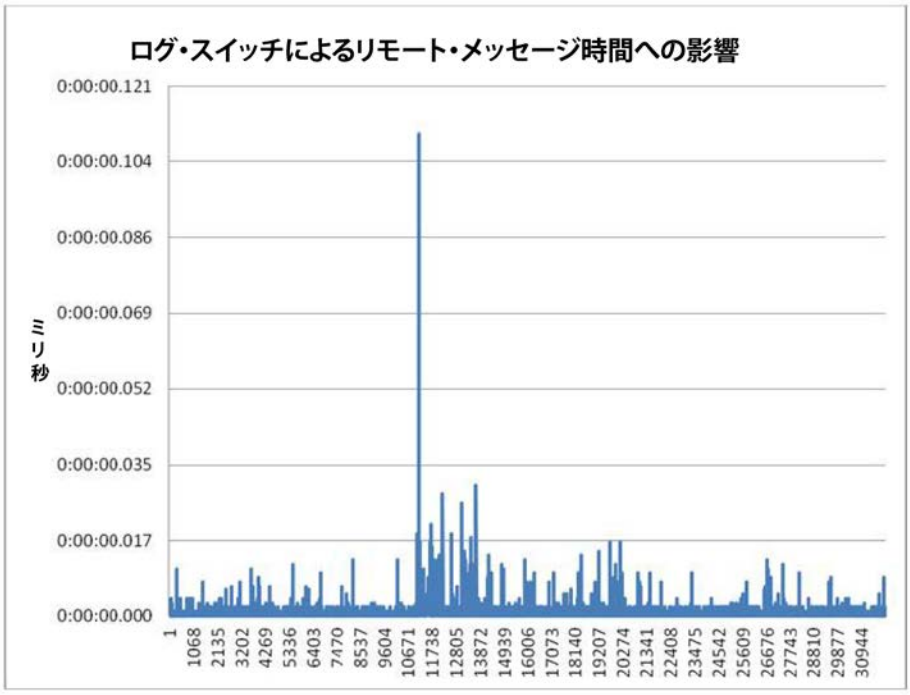

図3：ログ・スイッチによるリモート・メッセージ時間への影響

SYNCを有効にすると、他にどのような影響が出るでしょうか。

SYNCを有効にする場合は、コミット処理のため、通常のローカル書込みに加えてリモート書込み（スタンバイREDOログへのRFS書込み）を導入しています（Oracle Database 12cのFast Syncを使用しない場合）。このリモート書込みを使用すると、ネットワーク待機時間やリモートI/Oの帯域幅によっては、コミット処理時間が長くなる可能性があります。コミットの処理時間が長くなっているため、LGWRの作業終了とコミット・リクエストの作業開始まで待機するセッション数が増えます。つまり、アプリケーションの同時実行性が増加しています。アプリケーションの同時実行性の増加は、データベースの統計情報と待機イベントを分析すれば分かります。表6の例を見てみましょう。

表6：アプリケーションの同時実行性が増加するSYNC転送の影響

SYNC	REDO速度	ネットワーク待機時間	AWRからのTPS	ログ・ファイル同期の平均（ミリ秒）	ログ・ファイル・パラレル書込みの平均（ミリ秒）	RFSランダムI/O	SYNCリモート書込みの平均（ミリ秒）	REDO書込みサイズ（kb）	REDO書込み
遅延	25MB	0	5,514.94	0.74	0.47	該当なし	該当なし	10.58	2,246,356
あり	25MB	0	5,280.20	2.6	.51	.65	.95	20.50	989,791
影響	0	-	-4%	+251%	+8.5%	該当なし	該当なし	+93.8%	-55.9%

上記の例では、syncを有効にすると、REDO書込み数は減少しても、各REDO書込みのサイズは増加していることが分かります。REDO書込みのサイズが増加したため、（ローカルとリモートの）I/Oの実行時間は増えることが予想されます。待機あたりの作業が増えているため、'ログ・ファイル同期'待機時間は長くなることが予想されます。ただしアプリケーション・レベルでは、トランザクションの速度や応答時間への影響はほとんど変わらない可能性があります。コミットあたりのセッション処理数がより多いためです。このため、SYNCの影響の測定は、データベース待機イベントに完全に依存するのではなく、アプリケーション・レベルで行うことが重要です（詳しくは、以下を参照してください）。**これは、アプリケーションへの実際の影響を把握するのに、ログ・ファイル同期待機イベントを使用すべきでない理由を示す好例でもあります。**

Oracle Database 11.2のSYNCパフォーマンスの評価方法

パフォーマンスを調べるには、ローカルのREDO書込み待機時間、書込みあたりの平均REDO書込みサイズ、および全体的なREDO書込み待機時間を計算することを推奨します。この操作を実行するには、次の待機イベントを使用できます。

- ローカルのREDO書込み待機時間 = 'ログ・ファイル・パラレル書込み'
- 書込みあたりの平均REDO書込みサイズ = 'REDOサイズ'/'REDO書込み'
- 全体的なREDO書込み待機時間（ローカルとリモート） = 'REDO書込み時間'/'REDO書込み' * 10
- フォアグラウンドで確認できる平均コミット待機時間 = 'ログ・ファイル同期'
- 1秒あたりのトランザクション
- REDO速度

Oracle 11.2データベースのAWRレポートからの統計情報は、表7を参照してください。ローカル・スタンバイへのSYNC転送は、1ミリ秒のネットワーク待機時間で有効化されており、SYNCが無効なベースラインと、パフォーマンス上の影響を比較しています。

表7：Oracle Database 11.2でのSYNCパフォーマンスの評価

メトリック	ベースライン - SYNCなし	SYNC	影響
REDO速度（K/秒）	3,689	3,670	変化なし
ログ・ファイル同期	0.43	2.45	+469%
ログ・ファイル・パラレル書込み	0.30	0.30	変化なし
TPS	438	429	-2.1%
全体的な書込み待機時間	0.42	1.87	+345%
合計REDOサイズ	3,418,477,080	3,450,849,356	+0.9%
REDO書込み	426,201	396,199	-7.0%
REDO書込みサイズ	8,020	8,709	+8.6%

この例で分かるとおり、SYNCの有効化による全体的なトランザクション/秒への影響は、アプリケーションで2%でしたが、データベースREDO速度の低下は1%未満でした。また、REDO書込み数は減っていますが、サイズは増えています。この原因は、同時実行性や、1回のコミットまたはREDO書込みで処理されるセッション数の増加です。スタンバイへの平均書込み時間は1.87ミリ秒でしたが、ローカルの平均書込み時間は0.5ミリ秒未満でした。つまり、リモート書込みの1.87ミリ秒のうち、スタンバイREDOログへの書込みを完了するのに、ネットワークに1ミリ秒、I/Oに残りの約0.87ミリ秒が費やされたことになります。

Oracle Database 12cのSYNCパフォーマンスの評価方法

パフォーマンスを調べるには、ローカルのREDO書込み待機時間、書込みあたりの平均REDO書込みサイズ、および全体的なREDO書込み待機時間を計算することを推奨します。この操作を実行するには、次の待機イベントを使用できます。

» ローカルのREDO書込み待機時間 ='ログ・ファイル・パラレル書込み'

» リモート書込み待機時間 = 'SYNCリモート書込み'

» 書込みあたりの平均REDO書込みサイズ = 'REDOサイズ'/'REDO書込み'

» フォアグラウンドで確認できる平均コミット待機時間 ='ログ・ファイル同期'

Oracle 12cデータベースのAWRレポートからの統計情報は、表8を参照してください。ローカル・スタンバイへのSYNC転送は、1ミリ秒のネットワーク待機時間で有効化されており、SYNCが無効なベースラインと、パフォーマンス上の影響を比較しています。

表8：Oracle Database 12cでのSYNCパフォーマンスの評価

メトリック	ベースライン - SYNCなし	SYNC	影響
REDO速度（MB/秒）	25	25	変化なし
ログ・ファイル同期	0.68	4.60	+576%
ログ・ファイル・パラレル書込み	0.57	0.62	+8.8%
TPS	7,814.92	6224.03	-20.3%
RFSランダムI/O	該当なし	2.89	該当なし
SYNCリモート書込みの平均（ミリ秒）	該当なし	3.45	該当なし
REDO書込み	2,312,366	897,751	-61,2%
REDO書込みサイズ（kb）	10.58	20.50	+93.8%

上記の例では、SYNCを有効にすると、ログ・ファイル同期待機の平均が大幅に増えていることが分かります。ローカル書込みは比較的一定ですが、ログ・ファイル同期の増加の最大要因は、SYNCリモート書込みの追加でした。SYNCリモート書込みのうち、ネットワーク待機時間がゼロであることは分かっているため、スタンバイREDOログへのリモート書込みの平均時間は2.89ミリ秒になります。プライマリとスタンバイで同じハードウェアを使用していたとすれば、これは緊急の問題であり、リモート書込みとローカル書込みの時間を同じにする必要があります。

上記の例では、スタンバイREDOログ（SRL）に複数のメンバーが含まれており、パフォーマンスの低いディスクグループに配置されていました。単一メンバーに減らして高速ディスクグループに配置してから再度テストしたところ、表9のような結果になりました。

表9：SRLを単一メンバーに減らして高速ディスクグループに配置した場合のSYNCパフォーマンス

メトリック	ベースライン - SYNCなし	SYNC	影響
REDO速度（MB/秒）	25	25	変化なし
ログ・ファイル同期	0.67	1.60	+139%
ログ・ファイル・パラレル書込み	0.51	0.63	+23.5%
TPS	7714.36	7458.08	-3.3%
RFSランダムI/O	該当なし	.89	該当なし
SYNCリモート書込みの平均（ミリ秒）	該当なし	1.45	該当なし
REDO書込み	2,364,388	996,532	-57.9%
REDO書込みサイズ（kb）	10.61	20.32	+91.5%

スタンバイのI/O速度の向上によって、全体的なログ・ファイル同期待機が短縮され、1秒あたりのアプリケーション・トランザクション速度が向上しました。

ログ・ファイル同期待機の診断

ログ書込み時間が長くなると、プライマリ・データベースのLGWRトレース・ファイルに次のようなメッセージが書き込まれます。

```
Warning: log write elapsed time 163ms, size 18KB
```

デフォルトでは、500ミリ秒より長いログ書込みの警告のみが出力されます。次のアンダースコア・パラメータを使用して、これらのメッセージのしきい値を小さくすることができます。

```
_long_log_write_warning_threshold=<some number of milliseconds>
```

上記の警告に関連するタイムスタンプを使用すると、REDOの送受信を追跡することで、スタンバイが根本原因であるかどうかを診断できます。次の追跡を設定します：

» プライマリとスタンバイで、event 16410 (dynamic): alter system set 16410 trace name context forever, level 16と設定します。

» スタンバイで、クエリv$managed_standbyを実行してRFSのPIDを特定し、event 10046 level 12と設定します。

上記のトレース設定を使用すると、メッセージIDを追跡することでリモート・ログ書込みごとの所要時間を追跡できます。たとえば、リモート・ログ書込みごとに次のメッセージを出力し、メッセージの下にタイムスタンプの送信を配置します。

```
Client sending SQLNET request data [kpurcs] oper='Write' flag=67111042 thrd=1
seq=1584 msgid=117468

*** 2014-12-02 08:46:34.598 5837 krsu.c
```

スタンバイ側で、上記で特定したmsgidを探せば、リモート・ログ書込みの受信を確認できます。

```
...Server received SQLNET data [krsr_rfs_dsp] oper='Write' flag=67111042 thrd=1
seq=1584 msgid=117468

*** 2014-12-02 08:46:34.598 826 krsr.c
```

RFSでSRLへの書込みが完了すると、RFSからプライマリに対して、次のメッセージでackが再送信されます。

```
...Server finished processing SQLNET data [krsr_rfs_dsp] oper='Write' flag=67111042

thrd=1 seq=1584 msgid=117468

*** 2014-12-02 08:46:34.761 1459 krsr.c
```

プライマリでは、RFSからのackをNSSが受信したことを確認できます。上部にタイムスタンプが付いた次のメッセージが出力されます。

```
*** 2014-12-02 08:46:34.761 6058 krsu.c

...Client received SQLNET call [kpurcs] response oper='Write' flag=67111042
thrd=1 seq=1584 msgid=117468
```

例

次に、追跡と、長時間のログ書込み（163ミリ秒）の分析方法の例を示します。

NSSトレース・ファイルを見ると、163ミリ秒のほとんど全部がネットワークかRFS内で費やされていることが分かります。

```
Client sending SQLNET request data [kpurcs] oper='Write' flag=67111042 thrd=1
seq=1584 msgid=117468

*** 2014-12-02 08:46:34.598 5837 krsu.c

*** 2014-12-02 08:46:34.761 6058 krsu.c

...Client received SQLNET call [kpurcs] response oper='Write' flag=67111042
thrd=1 seq=1584 msgid=117468
```

イベント10046と16410がRFSで有効な場合、上記のmsgidでは次のように表示されます。

```
...Server received SQLNET data [krsr_rfs_dsp] oper='Write' flag=67111042 thrd=1
seq=1584 msgid=117468
*** 2014-12-02 08:46:34.598 826 krsr.c
WAIT #0: nam='RFS random i/o' ela= 298 p1=0 p2=0 p3=2147483647 obj#=-1
tim=1417535194598944
WAIT #0: nam='RFS write' ela= 162381 p1=0 p2=0 p3=0 obj#=-1 tim=1417535194761203
...Server finished processing SQLNET data [krsr_rfs_dsp] oper='Write' flag=67111042
thrd=1 seq=1584 msgid=117468
*** 2014-12-02 08:46:34.761 1459 krsr.c
```

このため、上記のRFS書込みの経過時間が、163ミリ秒のうちのほとんどを占めています。

次の例は、ログ書込みが長い（138ミリ秒）のネットワークで費やされている時間を示しています。

NSSトレース・ファイルを見ると、138ミリ秒のほとんど全部がネットワークかRFS内で費やされていることが分かります。

```
Client sending SQLNET request data [kpurcs] oper='Write' flag=67111042 thrd=2
seq=1311 msgid=124705
*** 2014-12-02 08:46:49.879 5837 krsu.c
*** 2014-12-02 08:46:50.017 6058 krsu.c
...Client received SQLNET call [kpurcs] response oper='Write' flag=67111042
thrd=2
seq=1311 msgid=124705
```

RFS側では、ほぼ170ミリ秒に達するまで、msgidが受信されていませんでした。

```
...Server received SQLNET data [krsr_rfs_dsp] oper='Write' flag=67111042 thrd=2
seq=1311 msgid=124705
*** 2014-12-02 08:46:50.016 826 krsr.c
WAIT #0: nam='RFS random i/o' ela= 360 p1=0 p2=0 p3=2147483647 obj#=-1
tim=1417535210017013
WAIT #0: nam='RFS write' ela= 246 p1=0 p2=0 p3=0 obj#=-1 tim=1417535210017052
...Server finished processing SQLNET data [krsr_rfs_dsp] oper='Write' flag=67111042
thrd=2 seq=1311 msgid=124705
*** 2014-12-02 08:46:50.017 1459 krsr.c
```

Data Guard Fast Sync

Fast Syncは、Oracle Database 12cで使用できる新しいData Guard機能です。Fast Syncでは、宛先パラメータNOAFFIRMを使用できます。このパラメータを使用すると、スタンバイREDOログ・ファイルへの書込みが完了するまで待たなくても、スタンバイでREDOの受信が認識されるように指定できます。Fast Syncを使用すると、合計ラウンドトリップ時間からリモートI/Oが除外されるため、SYNC構成でのアプリケーション応答時間を短縮できます。また、スタンバイI/Oパフォーマンスの変動や外れ値が、アプリケーションの応答時間に影響することを防ぐことができます。Fast Syncを使用すると、プライマリとData Guard同期先の間の距離が伸びても対応することができ、地理的な保護が強化されます。

Exadataを使用したオラクルのパフォーマンス・テストによると、Fast Syncを構成した場合、プライマリ・データベースのスループットが4%向上しました。皮肉なことに、Exadata Smart Flash Logを使用したExadataシステムの高速I/Oの方が、パフォーマンス上のメリットが少ないという結果になりました。低速I/Oのシステムにデプロイされた本番データベースでは、より実質的なパフォーマンス上のメリットがあります。たとえば、NASストレージ搭載のx86製品にデプロイされた仮想マシンで、Oracle Databaseを使用したパフォーマンス・テストを行った場合、Fast Syncを使用するとプライマリ・データベースのスループットが12%向上しました（図4）。これは、仮想マシンの例では、合計ラウンドトリップ時間のディスクI/Oの割合が高くなるためです。

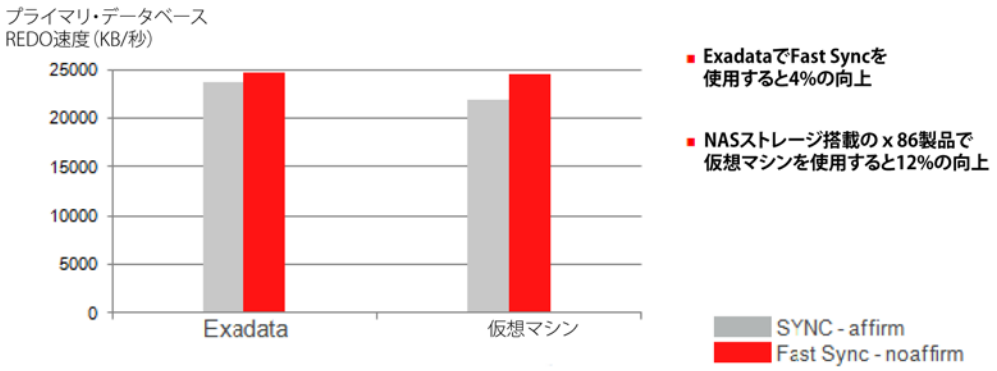

図4：パフォーマンスの比較：FAST SYNCとSYNC

結論

Data GuardとActive Data Guardの同期REDO転送を使用すれば、データベース、クラスタ、サイトの停止や、広範な地域に影響を与える可能性がある自然災害やその他のイベントが発生しても、データ損失ゼロの保護が可能です。

同期転送と高速データベース・フェイルオーバー（管理者が手動で開始するか、Data Guardファスト・スタート・フェイルオーバー[4]を使用して自動的に開始）を組み合わせれば、データ損失ゼロの保護、ディザスタ・リカバリ、および高可用性を実現できる、唯一のOracleレプリケーション・ソリューションとなります。本質的に、すべての同期レプリケーション・ソリューションは本番データベースのパフォーマンスに影響を与えます。本書で説明したように、Data GuardとOracle Databaseの独自統合と、MAAベスト・プラクティスを組み合わせれば、最高のデータ保護と最適なパフォーマンスを実現できます。

[4] http://docs.oracle.com/database/121/DGBKR/sofo.htm#i1027843

Oracle Corporation, World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065, USA

海外からのお問い合わせ窓口
電話：+1.650.506.7000
ファクシミリ：+1.650.506.7200

ORACLE

CONNECT WITH US

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

Hardware and Software, Engineered to Work Together

Copyright © 2015, Oracle and/or its affiliates.All rights reserved.本文書は情報提供のみを目的として提供されており、ここに記載されている内容は予告なく変更されることがあります。本文書は一切間違いがないことを保証するものではなく、さらに、口述による明示または法律による黙示を問わず、特定の目的に対する商品性もしくは適合性についての黙示的な保証を含み、いかなる他の保証や条件も提供するものではありません。オラクル社は本文書に関するいかなる法的責任も明確に否認し、本文書によって直接的または間接的に確立される契約義務はないものとします。本文書はオラクル社の書面による許可を前もって得ることなく、いかなる目的のためにも、電子または印刷を含むいかなる形式や手段によっても再作成または送信することはできません。

Oracle および Java は Oracle およびその子会社、関連会社の登録商標です。その他の名称はそれぞれの会社の商標です。

Intel および Intel Xeon は Intel Corporation の商標または登録商標です。すべての SPARC 商標はライセンスに基づいて使用される SPARC International, Inc.の商標または登録商標です。AMD、Opteron、AMD ロゴおよび AMD Opteron ロゴは、Advanced Micro Devices の商標または登録商標です。

Oracle is committed to developing practices and products that help protect the environment