



IBM DB2 pureScale  
スケールアウト・テクノロジーの比較

2009年 10月

## 目次

目次.....	- 2 -
はじめに.....	- 3 -
DB2 pureScale とは？.....	- 4 -
DB2 pureScale はどこから来たのか？.....	- 5 -
アプリケーションに透過的なスケーラビリティのための DB2 pureScale.....	- 6 -
DB2 pureScale による高可用性.....	- 8 -
Oracle Real Application Clusters ( Oracle RAC ) との比較.....	- 10 -
高可用性 - pureScale vs. Oracle RAC .....	- 10 -
拡張性 - pureScale vs. Oracle RAC.....	- 13 -
シナリオ 1: 1つのノードがディスクからデータページを読むとき.....	- 14 -
シナリオ 2: ある1つのノードが、別のノードが読み取っているデータページを更新する とき.....	- 16 -
シナリオ 3: 2つのノードが、同じデータページ上にある別々の行をそれぞれ更新する とき.....	- 19 -
Oracle の専門家がスケールするアプリケーション開発について語っていること.....	- 21 -
サマリー.....	- 23 -
参考文献.....	- 23 -

## はじめに

経済状況が回復しつつある中、主要なビジネス・データにリアルタイムにアクセスできることは生き残りと成功のために重要な要素です。効率的な投資を行う上で、ビジネスに対する鋭敏な感覚と高度な可用性を備えた柔軟なインフラストラクチャをもっていれば、回復する経済成長のなかでビジネス・チャンスをつかむことができます。

ほとんどのソフトウェア会社では高可用性を説明する場合に「メインフレームと同じ」「99.999%」の可用性とといった内容のマーケティング・メッセージを使います。これらのキー・フレーズは、もともと市場のゴール

Availability	Downtime per Year
99.999%	5 minutes
99.99%	50 minutes
99.9%	8 hours, 20 minutes
99%	3 days, 11 hours, 18 minutes
95%	18 days, 6 hours
90%	34 days, 17 hours, 17 minutes
85%	54 days, 18 hours

ド・スタンダードとなった DB2 for z/OS の高可用性を表現するメッセージです。

昨今、可用性とは、コンポーネントのエラーを回避して処理を継続し、通常のトランザクション処理を回復するということだけを意味しません。

もし、サービス・レベル・アグリーメント(SLA)が期待されるクエリーの応答時間を数秒と規定しており、サーバーが1分以内に結果を返すとなっている場合は、可用性の問題となります。システムが利用可能であるためには、トランザクションが処理できるだけでなく、SLA に定義された期間内にサービスを提供する必要があります。

例えば、ビジネスサイクルによる季節変動が、スケールの観点で可用性に影響を与えるならば、パフォーマンスの変化に対してアプリケーションを変更せずにリソースを追加するために、真に「透過的な」可用性を支えるアーキテクチャが必要となります。

「透過的な」というのが重要なポイントです。キャパシティを追加するとき、アプリケーションはクラスターの状況を意識する（ノード間の競合をさけるために、どのノードに何のデータがあるかを意識する）必要をなくすべきです。そこそこのスケーラビリティを得るために、このような複雑なアプリケーションを作成するために投資することは、今日のビジネス状況では許されません。

まず第一に、クラスターを意識したアプリケーションは、データ量による変更とその変更の反映が必要となります。クラスターを意識したアプリケーションは、クラスターの成長にあわせてコードの変更が必要となるだけではありません。変更内容をテストし、品質保証のためのプロセスを経て、配布し確認をしなければなりません。そのため、社内の調整に数週間の時間を取られ、必然的に他に有効活用可能なリソースを消耗させることとなります。

メインフレーム以外の分散プラットフォームにおけるデータベースのトランザクションに関するスケールアウトのオフリングは、時代遅れのアーキテクチャと（例えば、増加す

るオーバーヘッドのような)スケールに対する期待はずれの障害を持っており、SLA を満たすことができないことがあります。

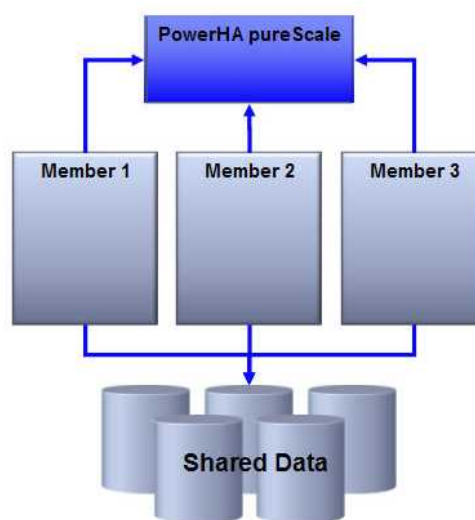
IBM DB2 pureScale テクノロジー (以降 DB2 pureScale) は現在と将来のビジネスニーズに応える可用性を提供し、アプリケーションに透過的なスケールアウトを可能とします。DB2 pureScale のアーキテクチャは、IBM Power Systems サーバーとストレージの統合により、高い価値のあるソリューションを提供できます。

いままでは、「メインフレームのような」はマーケティングのためのキャッチフレーズでした。DB2 pureScale は初の透過的なスケールアウトのアーキテクチャを分散プラットフォームで実現します。この文書では、DB2 pureScale テクノロジーを紹介し、どのようなテクノロジーか、どこから来たものか、どのように高可用性についての独自の優位点を持ち、アプリケーションに透過なスケールアウトを可能にするかを解説します。

## DB2 pureScale とは？

DB2 pureScale は、DB2 の新しいオプション機能で、複数のデータベースを「アクティブ - アクティブ」に設定された複数のサーバー上に展開し、高いレベルの可用性とスケラビリティを実現します。この構成により、それぞれのホスト (またはサーバー) で稼働している DB2 のコピーは、同時に同じデータに対してリードとライトのアクセスができます。DB2 データを共有する、ひとつまたは複数の DB2 サーバーを、データ共有グループと呼びます。データ共有グループに所属する DB2 サーバーは、そのグループのメンバーです。すべてのデータ共有グループのメンバーは同一のデータベースを共有します。現在のところ、データ共有グループのメンバーは最大 128 まで追加できます。

DB2 メンバーに加えて、powerHA pureScale コンポーネントがあり、ロックとデータページのグローバル・キャッシュ (グループ・バッファプール) の集中管理を行います。データ共有グループのそれぞれのメンバーは powerHA pureScale のコンポーネントを通じて直接情報を共有し、図に示すように、高速な InfiniBand ネットワークを通じて、それぞれのメンバーが、それぞれの接続を通じて、集中管理されたロック情報とキャッシュユーティリティにアクセスします。



## DB2 pureScale はどこから来たのか？

メインフレームと同様の可用性という引用を聞いたり読んだりするとき、その基準は DB2 for z/OS が提供するゴールド・スタンダードから取られています。実際のところ、System z サーバー上で稼働する DB2 for z/OS 以上の可用性を持ったデータベースソリューションは存在しません。

DB2 for z/OS の Coupling Facility (カップリング・ファシリティ 以降 CF) は、ロックとグローバル・キャッシュの一貫性を集中管理し、障害時の高速な回復を可能にします。このテクノロジーにより、DB2 for z/OS は SLA で要求される高い可用性を実現します。DB2 for z/OS は 99.999%の可用性を事実として提供し、ワークロードに対してシームレスに、またリニアにスケールすることで知られています。これはマーケティングの宣伝文句ではなく、長期間にわたってデータベースの可用性を提供し、業界をリードしてきた事実に基づくものです。おそらく、DB2 for z/OS に対する一番の賛辞は、Oracle 社の CEO ラリー・エリソンによるものです。

eWEEK.com 「In Larrys Own Words」より  
<http://www.eweek.com/c/a/Database/In-Larrys-Own-Words/2/>

“私は、Oracle以外のありとあらゆるデータベースを「大した事はない」と言い放ってきた。ただし、メインフレーム版DB2は別だ。メインフレーム版DB2は、第一級のテクノロジーのひとつだ。”

オラクル・コーポレーション CEO  
ローレンス・ジョセフ・エリソン(ラリー・エリソン)

DB2 for z/OS を特別なものとし、エリソンにこのような賛辞を送らせるものは何でしょうか？ DB2 for z/OS のユーザーには馴染み深いデータ共有のための「秘密の素」は、“Coupling Facility (CF)”です。CF は、DB2 for z/OS がリニアにスケールアウトすることを可能にします。また、ロックを集中管理して、ダーティ・ページのためのグローバルに共有されるバッファプールとして動作し、スケーラビリティと同時に高速なリカバリーを可能にしています。

DB2 pureScale テクノロジーは、この長年培われてきた DB2 for z/OS の CF テクノロジーを直接継承し、DB2 for z/OS がリードする高可用性とスケーラビリティのゴールド・スタンダードの利点を受け継いでいます。どうやって？ DB2 pureScale は、IBM powerHA pureScale コンポーネントと一緒に提供され、powerHA pureScale コンポーネントが、メインフレーム同様の集中ロック管理とグローバル・バッファプールのアーキテクチャを可能にします。

他社においても、共有ディスクのアーキテクチャは実装されており、最も有名なのは、Oracle Real Application Clusters (Oracle RAC)です。しかしながら、Oracle RAC が開発された当時、分散プラットフォームのテクノロジーでは、集中管理された共有キャッシュに効率的にアクセスすることはできませんでした。結果として、Oracle RAC のデザインは、DB2 for z/OS で確立されたテクノロジーをエミュレートしています。結果として Oracle RAC は分散ロック管理テクノロジーと分散キャッシュアーキテクチャを実装しました。Oracle RAC のアーキテクチャには、DB2 for z/OS のソリューションがスケールアウトする共有ディスクアーキテクチャを実装したときに持っていた簡潔な価値が欠けていました。一方、DB2 for z/OS と DB2 pureScale は同じ集中リソース管理の方法を採用し、スケーラビリティと可用性の複雑さを解決しています。詳細はこの文書の後半でご説明します。

真にアプリケーションに透過的にスケーラビリティと非常に高い可用性を提供するアーキテクチャは市場にただひとつしかありません。分散プラットフォームにおいては、近年利用可能となったハードウェア間の接続と InfiniBand 上の Remote Direct Memory Access (RDMA)により、ようやく DB2 for z/OS と同じロックの集中管理とバッファークッシングアルゴリズムが実装可能になりました。DB2 pureScale は、IBM ファミリーの中で進化し、分散プラットフォームにおいても業界において証明済みのテクノロジーを利用できるレベルに到達しました。

## アプリケーションに透過的なスケーラビリティのための DB2 pureScale

スケールアウトデータベース環境における真のコスト軽減のためのキーポイントは、アプリケーションに透過的なスケーラビリティです。透過的なスケーラビリティとは、データベースエンジンがスループットを維持し、データの分散配置を必要とせずに OLTP アプリケーションが十分なレスポンスタイムを得られることを意味しています。

データの分散配置というのは、ノード間で同一ページにあるデータをアクセスすると競合が発生することから、サーバーに接続するアプリケーションが必要とするデータについて置き場所を考慮する必要があるという意味です。ネットワークベースのメッセージング基盤に大きく依存してクラスター内でデータを共有するスケールアウトアーキテクチャにおいては、データの分散配置は非常に重要です。

データの分散配置に依存するスケールアウトアーキテクチャにおいて、十分なスケーラビリティを得るためには、アプリケーション開発者が洗練されたトランザクショナルなアプリケーションを開発する際に、クラスターを意識する必要があります。クラスターを意識したアプリケーションは、開発と配布にコストがかかります。また、クラスターの変更時

にアプリケーションも変更する必要があります。ベンダーによっては、アプリケーションを変更せずに利用可能だという主張をするかもしれません。しかし、クラスターを意識したデザインをしないと、どのようなアプリケーションもスケールしません。

アプリケーションに透過的なスケーラビリティとは、スケールアウトのアーキテクチャの有効性を得るために、アプリケーションがクラスターを意識する必要がないことを意味しています。DB2 pureScale は分散プラットフォームでは唯一、最新のネットワークとハードウェアアーキテクチャーを利用して、ロックとキャッシュの集中管理を有効に行います。ロック管理とグローバルキャッシュサービスのためのクラスター内のノード間のコミュニケーションを減らすため、DB2 pureScale は、powerHA pureScale Coupling Facility (以降 CF) を RDMA テクノロジーとともに使用しています。RDMA は、クラスター内のそれぞれのメンバーに CF 内のメモリーへの直接アクセスを許し、CF のメンバーそれぞれのメモリーに対する直接アクセスを可能にします。例えば、クラスター内のあるメンバー (メンバー 1) が、自分のローカル・バッファプールにないデータページを読みたいとします。DB2 は、エージェント(またはスレッド)をアサインしてこのトランザクションを処理します。エージェントは RDMA を使って CF のメモリーに直接書き込みを行い、該当ページのアクセスが必要なことを通知します。もしメンバー 1 が読みたいページがすでに CF のグローバル・バッファプールに存在する場合、CF はそのページをメンバー 1 のメモリーに直接配布するため、メンバー 1 のためにディスクからデータを読み出すための I/O 処理は発生しません。RDMA が使えるので、メンバー 1 のエージェントは単にリモートサーバーの memcopy(メモリー コピー)をコールするだけで、コストのかかるプロセス間コミュニケーションのためのコールやプロセッサのインタラプト、IP スタックのコールなどをする必要がありません。非常に簡単に言えば、pureScale では、メンバーのエージェントがローカルのメモリーコピーを行うのと同様に、同じ処理をターゲットとなるリモートマシンのメモリアドレスに対しても行うことができます。これらの単純なメモリーコールとバッファプールとロック管理の集中管理を行うことによって、アプリケーションはデータを実際に保持しているメンバーと会話する必要がありません。クラスターのサイズによらず、グローバル・バッファプールからデータページを受け取るので、クラスターのどのメンバーにとっても効率的です。ほとんどの RDMA コールは非常に高速なので、DB2 プロセスは CF からレスポンスを待つ間 CPU を明け渡す必要がなく、タスクを完了するまでにリスケジュールを行う必要がありません。例えば、CF に対して 1 行の更新を行うことを通知する(このとき X ロックが必要となります)には、メンバーのエージェントは Set Lock State(SLS)リクエストを CF のメモリーに直接ロック情報として書き込みます。CF は、他のクラスター内のメンバーがこの行の X ロックを取得済みでないことを確認し、リクエストしたメンバーのメモリーにロックの許可を直接書き込みます。この SLS に必要なやりとりは、15 マイクロ秒以下であり、そのためエージェントは CPU を明け渡す必要がありません。

エージェントは他のスケールアウトアーキテクチャで必要となる IP のインタラプトを待つことなく、(不必要なコンテキストスイッチを避けて)処理を継続することが可能です。ロングランバッチのトランザクションのような特別な操作のためには、DB2 エージェントが CPU を明け渡すほうが有効であると判断し、自動的にダイナミックに CPU をリリースします。

DB2 スケーラビリティのもうひとつの重要なポイントは、DB2 pureScale に組み込まれたクラスタメンバー間のロードバランスの機能が、アプリケーションに透過的なスケラビリティとともに提供されることです。アプリケーションはクラスターを意識しなくても、このロードバランスの機能を利用できます。DB2 for z/OS のデータ共有機能と同一のクライアント側のドライバが DB2 pureScale でも使用できます。

## DB2 pureScale による高可用性

スケールアウトのアーキテクチャは、キャパシティ追加のためにのみ提供されるものではありません。この種のアーキテクチャは、コンポーネントに障害が起きてもトランザクション処理を継続するシステムを構築する場合、可用性の向上にも役立ちます。

DB2 pureScale は、分散プラットフォームにおける他のオフリングに比較して、可用性を新しいレベルに引き上げます。DB2 pureScale は、リカバリーの必要なくすべてのデータページへのアクセスを提供し、I/O オペレーションを行うことなくリカバリーが必要なページを特定します。これもまた CF による集中管理によって可能となるユニークな能力による重要なイノベーションです。メンバーが自分のバッファプールにページを読み込むとき、CF は毎回その情報をトラックしています。メンバーがページの 1 行を更新するとき、同様に CF はその情報をトラックします。ダーティー・ページは更新を行うメンバーによって CF のメモリーに直接書き込まれます。このプロセスによって、クラスター内の別のメンバーが変更されたページを読み込もうとすると CF から直接更新された内容を得ることができます。さらに重要なのは、リカバリーの観点で、もしメンバーの誰かに障害が発生した場合、CF は、障害が発生したメンバーが書き換えようとしていたページのリストを持っており、コミット済みでまだディスクに書き込まれていない情報を持っています。

リレーショナルデータベースマネジメントシステム(RDBMS)のリカバリープロセスでは、まずコミット済みのトランザクションの処理をやり直します。コミットされたトランザクションによって更新されたページがディスク上に反映されていることを保証する必要があります。(このプロセスは REDO リカバリーとして知られています。)さらに、データベースサーバーは、障害以前にコミットされていないディスクに書かれていないデータの変更を行おうとしている仕掛りのトランザクションを元に戻す必要があります。(このプロセス

は、UNDO リカバリーとして知られています。)

共有ディスク・クラスターでは、クラスターの他のノードがリカバリー処理を完了していないページをディスクから読み込みや更新を行わないことが必須です。ページのリカバリーは、該当ページに対して新しいトランザクションによる更新が行われる前に済んでいなければなりません。CF が素晴らしいのは、問題が発生したノードがどのページを処理中だったかを知っていて、ダーティー・コミットを行ったページをすでにバッファープール上に持っていることです。そのため DB2 pureScale は、リカバリーが必要なページを特定するために、他のメンバーのトランザクション処理を止める必要がありません。他のアーキテクチャでは、ロック情報を分散して管理しているために、リカバリーが必要な情報を特定するために多くの処理時間が費やされます。(詳細は次章で説明します。)

DB2 pureScale におけるリカバリープロセスは概要レベルの説明をすることは簡単です。それぞれのメンバーはアイドルのプロセスを持っており、障害発生時にそのプロセスを利用することができます。メンバーに障害が発生してリカバリープロセスが呼び出されるとき、すでにプロセスが存在しているので、OS はプロセスの起動やメモリーのアロケーションに貴重なシステムタイムを割く必要がありません。リカバリープロセスは CF が持っているダーティー・ページの自分自身のローカル・バッファープールへのプリフェッチを即座に開始します。

ほとんどのリカバリー処理は、I/O を必要としません。リカバリーに必要なページは既に CF で集中管理されているバッファープールにあるからです。さらに、ページのプリフェッチは負荷の軽い RDMA を通じて高速に行われ、CF と他のリカバリメンバーの間で効率的に転送が行われます。このとき、他のメンバーはアプリケーションからの要求を受け続けることができます。リカバリーが必要ないデータやページが必要な場合は、トランザクションも継続できます。同様に、CF がディスク上のどのページがクリーンでどのページがリカバリーが必要かを正確に把握しているため、ディスクからページを読み込むこともできます。リカバリープロセスは、問題が発生したメンバーのログファイルを読み、更新を REDO または UNDO するために必要なトランザクションを実行します。

典型的なトランザクション処理のワークロードでは、メンバーに障害が発生してから、障害が発生したノードが更新しようとして仕掛りになっているページに次のトランザクションがアクセスするまで、20 秒かそれ以下の時間しかかかりません。他社のデータでは検知の時間を除外している場合がありますが、この時間には障害を検知するための時間も含まれています。メンバーに障害が発生してもその他のデータベース上のページはすべて利用可能となります。

さらに、PowerHA pureScale クラスター・アクセラレータは冗長構成が可能です。DB2 pureScale では、CF を 2 重に持つことが可能となっており、ロックと共有キャッシュの情報をプライマリの CF に問題が発生した場合に備えて、2 つの別々の場所に持つことができます。

## Oracle Real Application Clusters ( Oracle RAC ) との比較

これまでに述べたように、中央集約されたロック制御とバッファプールマネジメントによって、DB2 pureScale は類似製品に比べて明確なスケーラビリティと可用性の利点を持っています。さらに細部の違いを探ることで、Oracle Real Application Clusters (RAC) と比較してゆきます。

### 高可用性 - pureScale vs. Oracle RAC

システムの可用性における DB2 pureScale の優位性を例証するために、3 ノードのインスタンスが稼働していてその内 1 ノードがダウンした場合の例で考察してみます。そこで、Oracle RAC の 3 ノードクラスター環境で必要となるリカバリープロセスと対比します。両製品とも、ノード障害を検知するクラスターマネージャーを持ち、ソフトウェアもしくはサーバー障害を瞬時に検知することができます。そこで、ここでは障害検知時間よりむしろ、障害回復処理に焦点を当てることにします。DB2 pureScale は、フェイルオーバーの誤動作を避けつつ、ソフトウェアとハードウェアの障害を両方とも瞬時に検知する革新的な方法を採用していると言えます。

最初に、Oracle RAC の場合を見てみましょう。Oracle RAC では、各データページ( Oracle ではデータブロックと呼びます ) は、クラスター内のいずれかのインスタンスがマスターになります。Oracle は分散ロックメカニズムを採用しているため、クラスター内の各インスタンスは、自分がマスターになっている各データページに対するロック要求を管理する責任があります。ノード障害が発生すると、障害ノード上のデータページは、RAC がクラスター内の稼働ノードに所有権を割り当てる目的でロック情報を再配分している間、一時的に孤立することになります。この *Global Resource Directory (GRD)* の再配分が発生している間、データページへのいかなるロック要求はおろか、いかなる読み込み要求も一時的に抑止されます。アプリケーションは、稼働しているノード上では継続して実行できますが、この間はいかなる I/O 処理も新たなロック要求も行えません。これによって、多くのアプリケーションはフリーズすることになります ( 図 1 )。

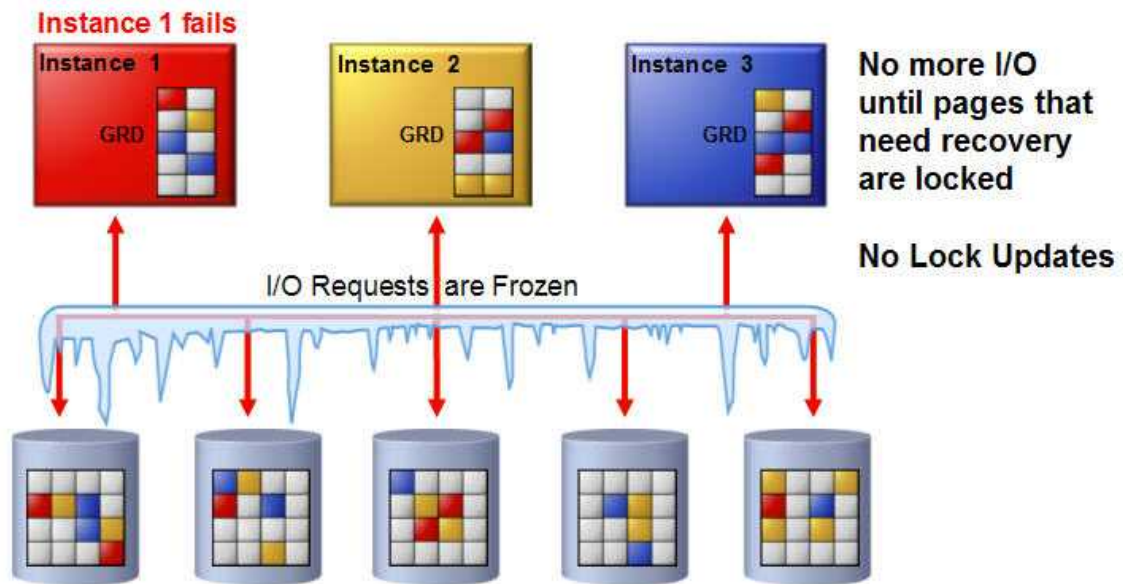


図1 – Oracle RACのリカバリープロセス中におけるI/Oフリーズ

RACのノードリカバリープロセスの2番目のステップは、回復を必要とする全データページをロックすることです。これは、先ほど説明したGRDフリーズが開放される前に行われなければなりません。もし、排他的なページロックが獲得される前に、あるインスタンスがディスクからデータページを読み込んでしまうと、障害のあったインスタンスが行った更新が失われることとなります。リカバリーを行うインスタンスは、障害インスタンスのREDOログファイルをファースト・パス読取りし、回復が必要な全てのページをロックします(図2)。ログファイル(およびリカバリーの必要なページも可能性があります)が、どの稼働しているノードのメモリー上にも存在しないかもしれないため、この方法ではかなりの量のランダムI/O処理が必要とされる可能性があります。

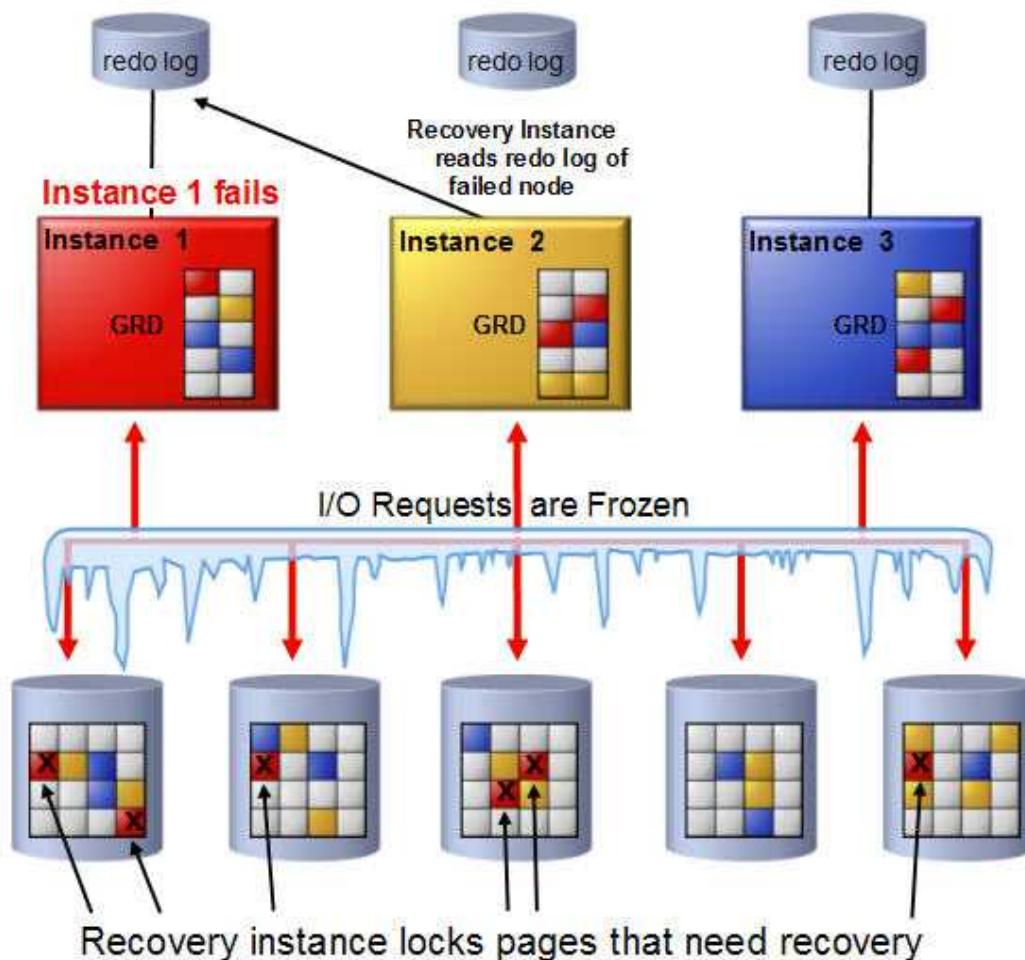


図2 – Oracle RACのファースト・パス・ログ読取り

GRD フリーズが解除されると、待たされていたアプリケーションは実行が再開できるわけですが、その前にリカバリーインスタンスによるこれらすべての I/O 処理と、適切なページのロックが完了しなければなりません。障害が発生したノードがその時点に実行していた処理量に応じて、このリカバリープロセスが完了するまでに数十秒から長い場合は1分程度までかかります。この GRD フリーズおよび、この間あるいは新規ロック要求が I/O 処理が行えないか、新規ロックが要求されるという事実は、この文章の巻末に参考文献として載せた Oracle RAC に関するいくつかの書籍に記載されています。

では、DB2 pureScale における実装を見てみましょう。対照的に、DB2 pureScale はクラスター内でグローバルのフリーズは発生しません。CF は、どのノードがダウンしたときにどのページにリカバリーが必要なのかを、常に意識しています。もしあるノードがダウンしても、クラスター内の他の全てのノードは、トランザクションを実行し I/O 処理を続けることができます。リカバリーが必要なページに対するアクセスだけは、リカバリー

プロセスによる障害ノードからのクリーンアップ処理が行われている間ブロックされます (図3)。

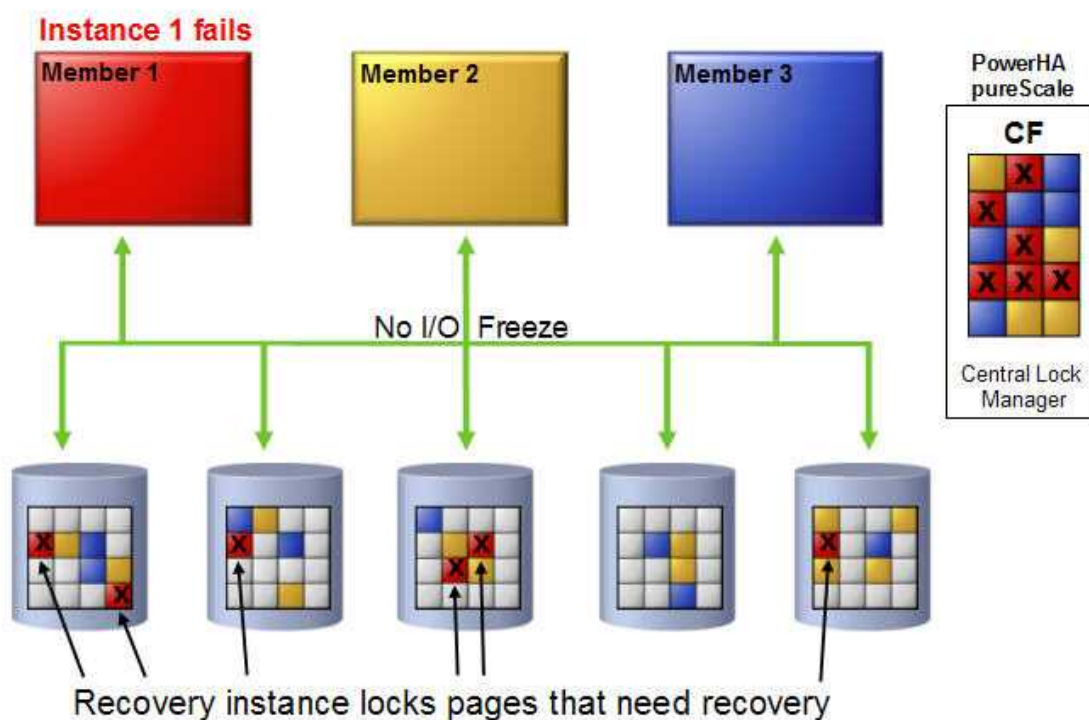


図3 - pureScaleによるリカバリープロセス

#### 拡張性 - pureScale vs. Oracle RAC

DB2 pureScale と Oracle RAC の拡張性の違いを明確に説明するために、いくつかの例を通してどのように互いのノードとのやり取りを行うかを示します。そのために、pureScale と Oracle RAC が下記のシナリオにおいてどのように動作するかを説明します。

1. ある1つのノードが、どのノードも保持していないページを読むとき
2. ある1つのノードが、別のノードが読み取っているページを更新するとき
3. 2つのノードが、同じデータページ上にある別々の行をそれぞれ更新するとき  
(これは一般的にホットページアクセスとして知られている)

**補足:** 以下の例では、Oracle の場合はサーバプロセスとして、DB2 の場合はエージェントプロセスとして、それぞれ動作しているアプリケーションについて言及します。

Oracle RAC の場合、クラスター内のノード間通信を制御するプロセス群があります。これらのプロセスは、グローバル・ロック要求 (LMD プロセス) や、グローバル・キャッシュ要求 (LMS プロセス) を処理します。簡潔にするために、ここでの例では、これらのプロセスの集まりをグローバル・キャッシュ・サービス (GCS) と呼びます。しかし、このプロセス間の通信によって発生するオーバーヘッドが、これから述べるよりもっと大きいということを知る必要があります。なぜなら、ロックやキャッシュ制御のために別々のプロセスが存在するからです。

#### シナリオ 1: 1 つのノードがディスクからデータページを読むとき

##### Oracle RAC

図 4 の例において、501 番のデータページへアクセスしようとしたサーバープロセスは、そのページが自ノードのバッファプール上に存在するかを最初にチェックします (図 4 の )。もし、該当ページが見つからなかった場合、サーバープロセスはそのデータページのマスターノードに問合せるために、GCS プロセスに対してプロセス間通信 (IPC) で要求を出します ( )。このことが、サーバープロセスが CPU を消費し、割り込みを処理するために CPU 上で潜在的に行われる GCS プロセスの再構成を目的としたコンテキストスイッチをもたらします。高レベルのコンテキストスイッチは、非常に CPU コストを消費します。GCS プロセスは、データブロックを要求するために、次にマスターノードに対して IP リクエストを送ります ( )。IP 通信は OS カーネル上で処理されるため、GCS プロセスは要求された情報をカーネルメモリ上に載せてから、リモートノードに要求を送るために IP スタックコールを行わなければなりません。たとえ InfiniBand ネットワークが使われていたとしても、Oracle は、未だに IP over InfiniBand かあるいは Reliable Datagram Sockets (RDS) を使用しています。たとえ InfiniBand 上であっても、ソケットプロトコルを使用することは、プロセッサ割り込みや IP スタックのトラバースなどによる負荷がかかります。

次に、リモートのマスター GCS プロセスは、割り込みを受信し、要求を処理するために CPU 上にスケジュールされます。他の全てのノードがバッファキャッシュ上に対象のデータページを持っているかを確認するために、GCS プロセスはチェックします。この例では、どのノードもそのデータページを持っていないため、GCS プロセスは IP メッセージを要求元に送り返し、ディスクからそのデータページを読む必要があることを教えます ( )。要求元ノードの GCS プロセスは、入ってきた IP リクエストを処理するために再び中断され、クラスター内でどのノードもそのデータページを持っていないことを知らせるために、順番に IPC 割り込みをサーバープロセスに送信します ( )。サーバープロセ

スは、ディスクからデータページを読み取りバッファークッシュに格納します( )。

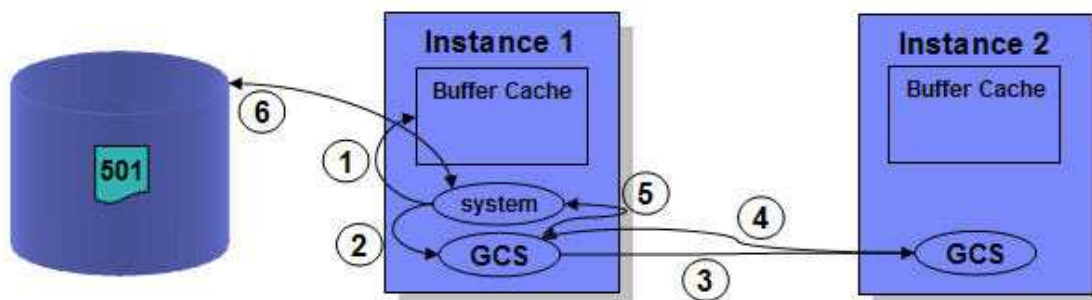


図4 - Oracle RACによるデータページの読取り

### DB2 pureScale

DB2 pureScale がより拡張性に優れていて、このようにデータの格納場所を探す必要がない理由は、同じ例を使って DB2 pureScale の動作を説明すると明確になります。最初のステップは Oracle RAC と同じです。エージェントプロセスは、最初に必要としているデータページが既にメモリー上にあるかを確認するために、自ノードのバッファープールをチェックします(図5の )。もし、そのデータページが見つからない場合、エージェントプロセスは CF に対する Read and Register (RaR) リクエストを、RDMA 経由で自ら送信します( )。これは、図4で示した Oracle RAC のケースとかなり違いがあります。DB2 pureScale では、エージェントプロセス自身が CF のメモリーを直接更新します(他プロセスへの IPC ではなく、割り込みもなく、カーネル経由の IP コールでもなく、コンテキストスイッチなども必要とせずに)。もし、エージェントプロセスがデータページ 501 番を読みたい場合、CF のメモリー上には「私はデータページ 501 番が必要で、バッファープールの 42 番スロットに格納したい。」というメッセージが書かれます。CF がこのリクエストを受け取ると、CF はそのデータページが存在するかグローバルバッファープール (GBP) をチェックします。そのデータページが見つければ、CF はそのデータページの内容を、要求したノードのメモリー上に RDMA 経由で直接書き込みます。そのデータページが見つからなければ、エージェントプロセスにディスクからデータページを持ってくることを指示するために、要求したノードのメモリー上に直接返答が書かれます( )。そして、エージェントプロセスは必要な I/O を実行します( )。

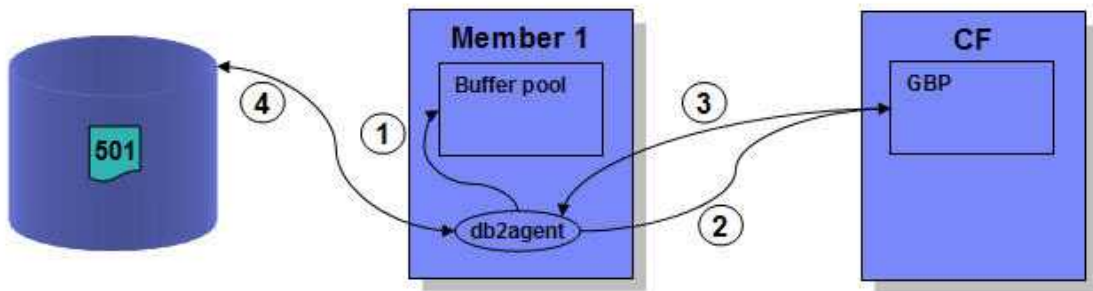


図5 – DB2 pureScaleによるデータページ読取り

RDMA を使用することで、エージェントプロセスが多くの場合 CPU を必要としないため、IP 通信より効率的です。エージェントプロセスは、実際にはデータの要求に memcopy コマンドを使用することになり、自ノードのローカルメモリから単純に読み込むこととなります (図 6)。

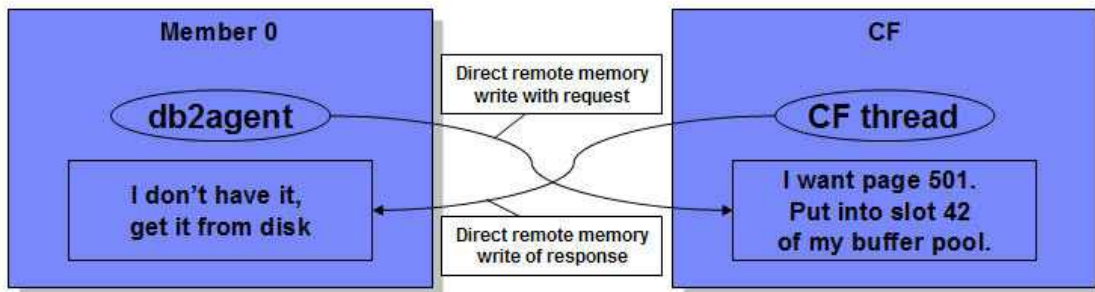


図6 –DB2 pureScale環境におけるRDMAリクエスト

シナリオ 2: ある 1 つのノードが、別のノードが読み取っているデータページを更新するとき

### Oracle RAC

この例では、3 ノード RAC クラスター環境において、インスタンス A が 501 番のデータページを更新しようとしていると仮定します。このデータページは、インスタンス B がマスターで、インスタンス C のバッファキャッシュ上に存在しています。このケースでは、初期のステップは同じです。明らかに、エージェントプロセスはデータページを更新しようとしていることを知らせるために、IPC リクエストをローカルの GCS プロセスに送ります (図 7 の )。GCS プロセスは、データページの更新を要求するために、IP リクエストをマスターノード (インスタンス B) に送信します ( )。このケースでは、マスターノードはそのデータページが既にインスタンス C によって読み取られていることを知っ

ているため、マスターノードは別の IP リクエストをインスタンス C に送信し、そのデータページへの読取りロックを開放してインスタンス A にそのデータページを渡すことを指示します( )。インスタンス C の GCS はこの IP メッセージを受信すると、処理を中断し、そのデータページの共有ロックを開放し、IP スタック上にデータページをコピーし、インスタンス A にデータページをネットワーク経由で送信します( )。一旦データページがインスタンス A の IP スタックを通じて受け取られると、そのノード上の GCS プロセスは、今そのデータページを更新のために排他モードで保持したことを知らせるために、別の IP メッセージをマスターノードに送り返します( )。もしクラスター内に 3 ノード以上あれば、マスターノードは、そのデータページを読取り用にローカル・バッファプール上に持っている全てのノードに対して、そのグローバル・ロックを開放しなければならないことを知らせます。そして、そのデータページを更新するために保持するノードが 1 つになります。

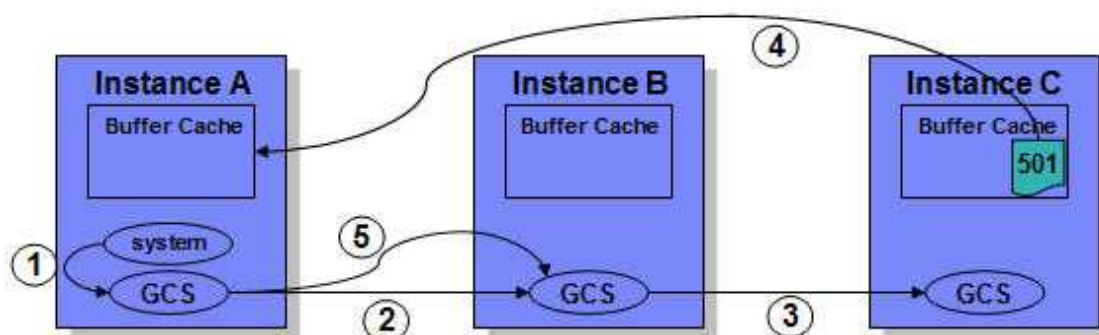


図7 - 1つのインスタンスが別インスタンスのキャッシュ上にあるデータページを更新するとき

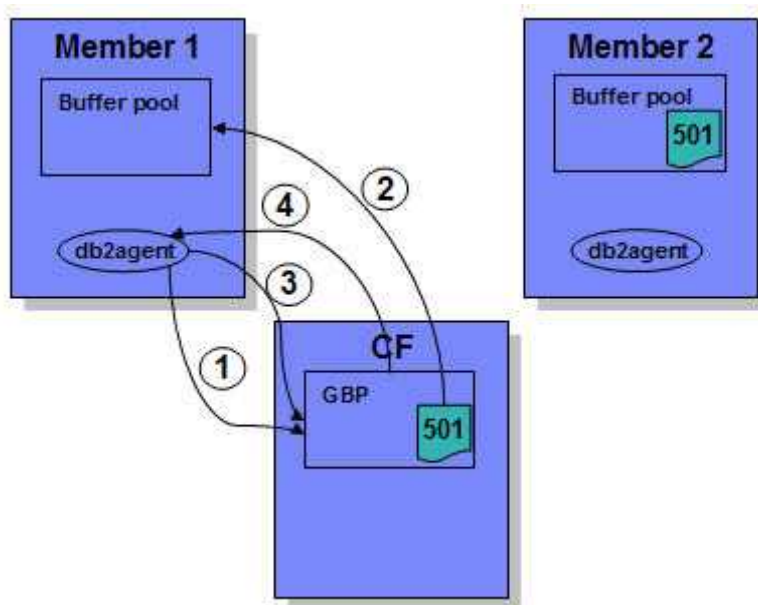
### DB2 pureScale

DB2 pureScale は、クラスター内で同じデータページの所有権を持つノード数が 2 ノードと 3 ノード以上の場合で、動作が異なります。メンバー 1 が、他のメンバーが既に CF に所有権を登録していて、既に中央集約されたバッファプール (GBP) に存在しているデータページを更新しようとしたケースを想定します。今度も、初期のステップは先ほどと同じです。エージェントプロセスは、ローカルのバッファプールをチェックし、そのデータページが見つからないときは、CF のメモリーにデータページの内容と格納先のバッファプールスロット情報を直接書き込みます(図 8 の )。このケースでは、CF はそのデータページを持っているので、CF はメンバー 1 のメモリー上にデータページの内容を直接書き込み( )、その時点でエージェントプロセスはデータページ上の対象行を処理できるようになります。この点で、CF は他のどのメンバーに対して何のアクションも通知する必要はありません。なぜなら、DB2 pureScale 環境では、そのメンバーがデータペ

ージ上を実際に更新するまで、データページ上に排他ロックを取得しないからです。

もし、このメンバー上のエージェントプロセスが変更したい行を見つけた場合、エージェントプロセスは CF に Set Lock State (SLS) コールを使用して通知します。SLS コールも RDMA 処理であり、メンバーが直接 CF のメモリー上に、ロックしたい行と取得したロックの種類を書き込みます( )。このケースでは、もしエージェントプロセスがある行を更新するならば、CF に対してその行の X ロックとそのデータページへの P ロックを取得することを知らせます(実際にはデータページに対するバイト単位の更新を行うことを知らせます)。クラスター内でこのデータページを今更新しているメンバーが他になく、他のどのメンバーもこの行の X ロックを取得していないならば、CF は要求元のメンバーのメモリー上を直接更新して、ロックが取得されたことを知らせます( )。競合しないロック要求のため、SLS コールに要する時間は約 15 ミリ秒以下でとても効率的である、しかもここでもエージェントプロセスは CPU 負荷を必要としません。

もしクラスター内の他のメンバーが同じページを更新しようとしている場合、実際にデータページが更新された後すぐに、そのメンバーは P ロックを解放することができます。しかし、もし他のメンバーがこのデータページを更新しない場合、そのメンバーは P ロックを持ち続け、トランザクションのコミット時点で P ロックを開放します( *Lazy* なロック開放と呼ばれます)。これらのロックは、ロックを要求するメンバーが他にない場合を除いて、トランザクション全体の過程に渡って取得されるわけではありません。他のメンバーがこのロックを要求したらすぐに、ロックを保持していたメンバーは、トランザクションが完了していなくてもそのロックを開放します。



## 図8 – メンバー1がグローバル・バッファプール上にあるデータページを更新するとき

シナリオ 3: 2つのノードが、同じデータページ上にある別々の行をそれぞれ更新するとき

### Oracle RAC

2つのノードが、同じデータページにある別々の行をそれぞれ更新するとき、Oracle RACを使用することでのデータの局所性の欠如が深刻な問題になり得ます。クラスター内でのホットページの特定方法や、データページにより少ない行を格納するかアプリケーションをパーティション化することによってこの問題を軽減する方法などについて、Oracle RACに関して発行されている書籍が複数存在します。Oracle RAC 環境の2つのノードが、同じデータページ内の別々の行を更新するときの動作を図9で例示しています。

注釈：この例では、GCS と IP 割り込みのオーバーヘッドについて、繰り返し言及することを控えています。あるノードから別のノードへの全てのリクエストは、この負荷の高いスタックを通らなければなりません。

インスタンス A が、インスタンス B がマスターであるデータページ内の行を更新しようとしています。このときインスタンス C は、同じページの行(インスタンス A が更新する行とは別)を更新しようとしており、インスタンス C はグローバル X ロックと共にそのデータページを既に取得しています。インスタンス A は、更新のためにそのデータページを取得することをインスタンス B に要求します(図9の )。インスタンス B は、ページロックを開放しデータページをインスタンス A に送信するようにインスタンス C に指示します( )。インスタンス C は、処理を中断しデータブロックをインスタンス A に送信し( )、排他的なグローバルページロックがインスタンス A によって取得されることをマスターノードに伝えます( )。そしてインスタンス A は、更新しようとする行を見つけるためにデータページ内の行を検索し始めます。しかし、インスタンス C 上のサーバープロセスは、まだ全ての行を処理し終えていなかったため、排他モードでデータページを取り戻す要求をするためにインスタンス B 上のマスターにメッセージを送信します( )。マスターは、インスタンス A を遮り、そのデータページの排他ロックを開放してインスタンス C に再び渡すように伝えます( )。インスタンス A はこれに従い、ネットワーク経由でデータブロックを送信し( )、その時点でインスタンス C は、マスターにデータページを取得したことを知らせます( )。

ここで、各インスタンスがデータページの各行を見直してそれぞれ更新するまで、2つのインスタンスはネットワーク越しにデータページの受け渡しを行うこととなります(図9には簡略化のため記載していませんが、メッセージとデータページをインスタンス間で往

復させるサイクルが続きます。)

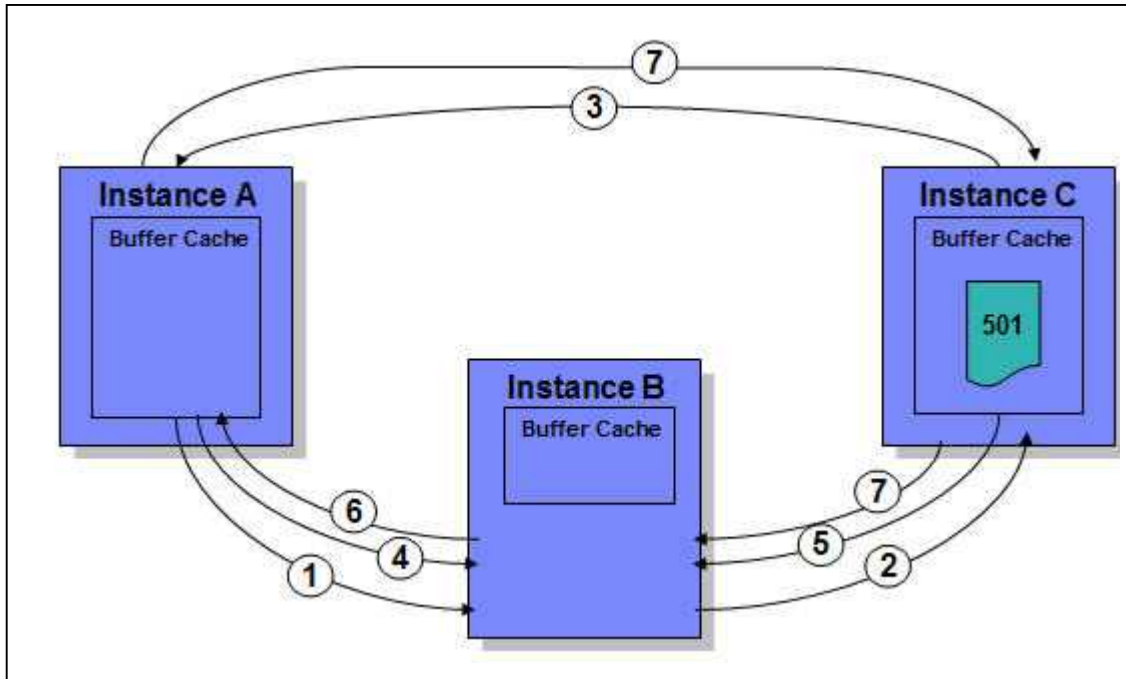


図9 – Oracle RACによるホットページへのアクセス

### DB2 pureScale

この同じシナリオは、DB2 pureScale 環境では全く違ってきます。メンバー1 が現在 CF 上にある行を更新しようとしていて、メンバー2 も同じデータページの別の行を更新しようとしている、先ほどと同じ例を想定します。

このシナリオでは、メンバー1 は CF にデータページを要求し( 図 10 の )、前回の例と同様に、CF からメンバー1 のメモリーに直接書き込まれます( )。各メンバーは、たとえ両メンバーがそれぞれ行を更新しようとしていても、同時にそのデータページ上の行を読み込むことができます。両メンバーは更新したい行を見つけると、各メンバーは更新対象行の X ロックと該当データページの P ロックを CF から要求します( )。最初に要求したメンバー(この例ではメンバー1)にロック権が与えられ、データページへの更新が許されます( )。もう一方のメンバー(この例ではメンバー2)には、以下の2つのうちのどちらかのイベントが発生するまでの間、データページ上の行を読み続けることが許されます。

メンバー1 がトランザクションをコミットしたとき。メンバー1 は新しいバージョンのデータページを CF のメモリー上に送り、直後に CF はメンバー2 に対して現在持っているデータページが無効であることを知らせます。CF は、新しいデータページのコピーをメ

ンバー2に送ります（全て RDMA 経由）。

メンバー2が更新したい行を見つけたとき。メンバー2はロック要求をCFに行い（ ）、CFはメンバー1にPロックを解放するように伝えます。メンバー1は、RDMA 経由でのメモリーコピーを使って、更新したデータページをグローバル・バッファプールに送ります（ ）。そしてCFは、新しいバージョンのデータページをメンバー2のメモリー上にコピーし（ ）、データページの更新要求を許可します。

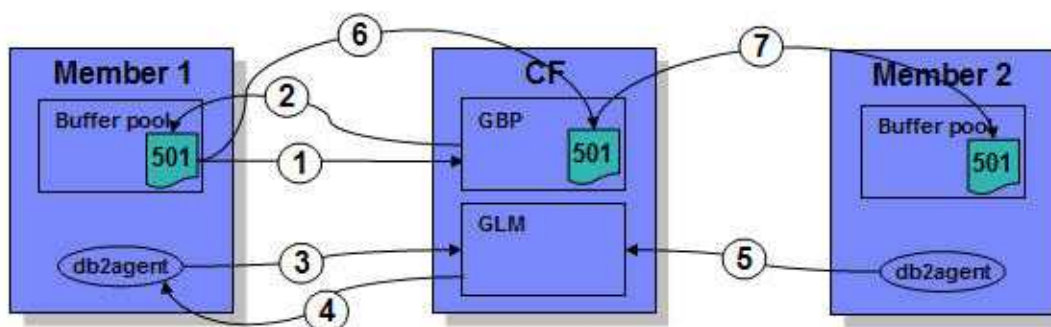


図10 – DB2 pureScaleによるホットページへの更新

Oracle RAC との主な違いは、Oracle RAC はデータページ上の1つの行を更新しようとしているときでもページロックが獲得されることです。一方 DB2 pureScale では、ページロックは実際に行の更新が行われるときのみ獲得されます（そのためにページがサーバー間を行き来します）。これによって、多重更新時における同時実行性が向上することになり、データの局所性について心配する必要がなくなり、それゆえアプリケーションを変更せずに高いレベルのスケールビリティが可能になります。

Oracle の専門家がスケールするアプリケーション開発について語っていること

Oracle RAC では、アプリケーションがアクセスするデータページを管理するノードが、アプリケーションが接続しているノードである場合に、最も拡張性が得られます。クラスターの規模が大きくなると、マスターノードが接続先のノードである可能性が低くなる点を注意すべきです。例えば、2 ノードクラスターでは、アプリケーションの稼働するノードがデータページのマスターである可能性は、50%です。これが10 ノードクラスターでは、10%に低下します。同様に、より良い拡張性を得るために、ある一組のデータページを更新するトランザクションは、別の組のデータページをアクセスしている別のノードで稼働するトランザクションとは、確実に隔離すべきです（ホットページの問題を回避する

ために)。これは、データを局所的にアクセスするためのアプリケーションとデータベースのパーティショニングとして知られています。

データを局所的に得るためにアプリケーションをパーティション化する技術は、日本オラクル(株)データベースグループのシニアエンジニアである「Tsutsui Tomonari」氏が行ったプレゼンテーションに記載されている。Tomonari氏は、Oracle Unbreakable Summit (session C-4) において「Oracle Real Application Clusters 10g Best Practices」というタイトルのプレゼンテーションを実施している。このプレゼンテーションの中で Tomonari氏は、新規アプリケーションにとって、Oracleの拡張性に対するベストプラクティスは、競合を減らし、別々のノードから同じテーブルをアクセスすることを避けるため、可能な限りアプリケーションをパーティション化することであると述べている。さらに、ノード間のデータブロック競合を減らすため、データブロック内の行数を減らしホットページを避けるために表をパーティション化すべきであるとも述べている。さらには2008年2月に、Oracle-I フォーラムに以下の書込みがあった。

「私は以下を推奨します。

ブロックレベルの競合を避けるために、アプリケーションをパーティション化するよく使われるオブジェクトには、より小さいブロックサイズを使用する (3)」

同様に、2009年7月8日に José Valerio によってブログへの書込みがあり、ノード間通信とその通信による待ちイベントに関する問題が議論されている。この書込みで Valerio氏は、アプリケーションの変更が必要となるであろう「コミットの頻度を抑える」と、「ブロックレベルの競合を避けるためにアプリケーションのパーティション化 (4)」を推奨している。Valerio氏は単なる Oracle 専門家ではなく、実際に元 Oracle 社員として Oracle US にて RAC 開発チームで働いており、Oracle 社では「Oracle Technologies, RAC Specialist」であった (5)。

クラスターを意識したアプリケーションの開発費用は、データの局所性なしにスケールするアプリケーションの開発費用より明らかに高額である。加えて、3ノードでスケールするためにパーティション化されたアプリケーションは、データベースが4ノードになったときに変更が必要になります(それゆえ再テストと再デプロイが必要です)。

DB2 pureScale では、クラスターのサイズに関わらず中央集権化された CF が通信します。非常に効率的なメモリー間コピーのアーキテクチャを使用し、真のグローバル・キャッシュ内にホットページを集約することにより、DB2 pureScale は透過的なアプリケーションの拡張性をもたらし、分散プラットフォームにおけるアプリケーション開発と改修のコストを削減します。

## サマリー

新しいハードウェアアーキテクチャーの活用により、DB2 pureScale はかつて DB2 for z/OS でのみ利用可能であった中央集権化されたロックとキャッシュの性能を、分散プラットフォームにもたすることができる。このハードウェアとネットワークの利用は、高いレベルの同時実行性とオーバーヘッドの劇的な削減につながり、その結果高いレベルの拡張性をもたらします。加えて、中央集権化されたロックとページキャッシュによって、DB2 pureScale はいずれかのメンバーがダウンした際にどのデータページにリカバリーが必要であるかを常に認識します。したがって、障害時において、リカバリー不要な全てのデータは継続的にアクセス可能であり、ダウンしたノードによって更新中であったデータページは、システムには既にわかっているため、より早くリカバリー可能である。

高いレベルの可用性が求められ、水平レベルの拡張性がコストメリットに繋がるアプリケーションにとって、DB2 pureScale はこれらのニーズを満たす解決策をもたらし、市場で既に証明された歴史を持っています。

## 参考文献

- 1) eWeek article <http://www.eweek.com/c/a/Database/In-Larrys-Own-Words/2/>
- 2) Sample of publications that clearly illustrate the Global Resource Directory Freeze:
  - ・ . Oracle Real Application Clusters – ISBN 978-1555582883 – page 493
  - ・ . Oracle 10g Real Application Clusters Handbook – ISBN 978-0071465090 – page 182
  - ・ . Oracle 10g RAC: grid, services & clustering – ISBN 978-1555583217 – page 265
- 3) Oracle-L forum posting  
<http://www.freelists.org/post/oracle-l/gc-current-block-busy-wait-in-Concurrent-BATCH-Processes-Run,2>
- 4) Blog posting at <http://jose-valerio.com.ar/blog/?p=196>
- 5) About page for José Valerio [http://jose-valerio.com.ar/blog/?page\\_id=2](http://jose-valerio.com.ar/blog/?page_id=2)

## IBM DB2 pureScale スケールアウト・テクノロジーの比較

2009年10月

© Copyright IBM Corporation 2009

IBM Canada  
8200 Warden Avenue  
Markham, ON  
L6G 1C7  
Canada

米国で作成

2009年10月

All Rights Reserved.

IBM、DB2、pureScale、Power、および z/OSは、世界の多くの国で登録された International Business Machines Corp. の商標です。他の製品名およびサービス名等は、それぞれ IBM または各社の商標である場合があります。現時点での IBM の商標リストについては、

<http://www.ibm.com/legal/copytrade.shtml> をご覧ください。

UNIXおよびUnixに関連する商標およびロゴは、The Open Group の米国およびその他の国における商標または登録商標です。その他の会社名、製品名またはサービス名はそれぞれ各社の商標です。本書に記載の製品、プログラム、またはサービスが日本においては提供されていない場合があります。日本で利用可能な製品、プログラム、またはサービスについては、日本 IBM の営業担当員にお尋ねください。

以下の保証は、国または地域の法律に沿わない場合は、適用されません。本書は特定物として「現存するまま」の状態を提供され、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任を負わないものとします。国または地域によっては、法律の強行規定により、保証責任の制限が禁じられる場合、強行規定の制限を受けるものとします。

この情報には、技術的に不適切な記述や誤植を含む場合があります。本書は定期的に見直され、必要な変更は本書の次版に組み込まれます。IBM は予告なしに、随時、この文書に記載されている製品またはプログラムに対して、改良または変更を行うことがあります。

この文書に含まれるいかなるパフォーマンス・データも、管理環境下で決定されたものです。そのため、他の操作環境で得られた結果は、異なる可能性があります。一部の測定が、開発レベルのシステムで行われた可能性がありますが、その測定値が、一般に利用可能なシステムのものと同じである保証はありません。さらに、一部の測定値が、推定値である可能性があります。実際の結果は、異なる可能性があります。お客様は、お客様の特定の環境に適したデータを確かめる必要があります。

IBM 以外の製品に関する情報は、その製品の供給者、出版物、もしくはその他の公に利用可能なソースから入手したものです。IBM は、それらの製品のテストは行っておりません。したがって、他社製品に関する実行性、互換性、またはその他の要求については確認できません。IBM 以外の製品の性能に関する質問は、それらの製品の供給者にお願いします。

本書に含まれる情報は、情報提供のみを目的に提供されています。本書は、特定物として現存するままの状態を提供されるものであり、すべての明示もしくは黙示の保証責任を負わないものとします。

一般に入手可能な情報源から得た情報は2009年10月1日現在の情報で、変更されることがあります。この文書に含まれるすべてのパフォーマンス・データは特定の環境下で得られたものであり、例として提示されるものです。他の稼働環境では、パフォーマンスが異なる場合があります。製品性能に関する詳細情報については、それらの製品の供給者から入手して下さい。