4

データベース・システムにおける24時間ノンストップの手法

24-Hour Non-Stop Methods in Database Systems

日本アイ・ビー・エム システムズ・エンジアリング株式会社
第二システム・センター
ハイアベイラビリティー・システム部
グループリーダー
DB2 Advanced Expert
吉澤 剛士

Takeshi Yoshizawa
DB2 Advanced Expert
Group Leader
High Availability Systems
System Center No.2
IBM Japan Systems Engineering Co.,Ltd.

今日、コンビニエンス・ストアなどは24時間営業が当たり前になり、あるいは銀行のATMなども営業時間を延ばしています。さらにインターネットを用いたe-ビジネス環境では24時間いついかなるときでもアクセスできるのが当然です。
そのためシステムは、でき得る限りの分散化・2重化により、いずれかがダウンしても処理が続行できる仕組みになっています。しかしWebサーバーやアプリケーション・サーバー、ディスパッチャーなどとは異なり、データをつかさどるデータベースはそう簡単にはいきません。データの整合性を取るために、一元管理はいわば宿命となっているからです。
本論文ではデータベース・システムを中心に、ミドルウェアが提供する機能や、他社ベンダーのアーキテクチャーおよび取り組みなども交え、ノンストップ・システムへ向けての手法を検証し、さらに現状を踏まえつつ今後のノンストップ・システムへの展望を試みました。24時間連続稼働を踏まえたデータベース・システムの実現に向けて少しでも役立てば幸いです。

It has become common for convenience stores and other stores to remain open 24 hours a day, and the automatic teller machines in banks are also open much longer than they used to be. And of course the e-business environment allows for access 24 hours a day wherever one may happen to be through use of the Internet.
Because of this, the system is such that processing can be continued even in the event of a failure occurring through the greatest possible degree of decentralization and doubling of functions. But this isn't possible with data-handling databases, which differ from Web servers, application servers, dispatchers and firewalls. This is because unified, centralized control is absolutely essential in order to ensure the coordination of data.
In this paper we take a look at methods oriented toward non-step systems, centering on database systems and with consideration given to the functions presented by middleware and the architecture and efforts being made by other vendors. I have also examined the prospects for future non-step systems in light of their current status. I hope that this study will contribute to some extent to the realization of database systems oriented toward 24-hour continuous operation.

1. はじめに

e-ビジネスが普及して24時間連続稼働が当たり前の時代になると、SPOF（Single Point of Failure）となり得るデータベースの信頼性がますます重要になることは間違いありません。また、今までのような照会系がメインのシステムならともかく、e-ビジネスにおいて更新処理を扱うインターネット・システムでは障害によるデータの損害は致命的なものになりかねません。それ故に、障害が発生した場合に業務をどうやって継続させるか、すなわち24時間ノンストップで運用するかが大きなカギとなってきています。そうしたニーズに対応するために、多くのデータベース製品はIBMのDB2 Universal Database™（UDB）も含めて、バージョンを重ねるにつれて改良されてきています。しかしながらほとんどのデータベース製品は（特にUNIX®系においては）フォールト・トレラント専用に設計されてきたわけではありません。それを補うために幾つかのミドルウェアとの連携などが行われていますが、まだまだ完全ではないのです。

本論文ではIBMのデータベース製品にとらわれず、他社のデータベースへの取り組みなども交えながら、24時間ノンストップ・システムの現状と手法をまとめ、さらに今後の展望についても模索しました。

2. 24時間ノンストップのデータベース・システムに求められる要件

24時間ノンストップ・システムとは、可用性を常に提供するように構成された高可用性システムのことを指しますが、業務の形態によって期待される可用性の評価は大きく異なります。インターネットのメール・サーバーのようなミッション・クリティカルもしくはビジネス・クリティカルなアプリケーションになるほど、高い可用性が必要とされます。また1日24時間、常に稼働が求められる業務形態もあれば、株式市場のように市場が開いている時間帯に100%近い稼働率が求められるケースもあります。一般に、可用性の高さの基準には平均リカバリー時間（MTTR）と平均障害間隔時間（MTBF）の2種類が用いられますが、多くのシステムやお客様では、平均リカバリー時間（MTTR）をどれだけ縮めるか、そのためにいかにシステムを最適化するかに重点を置いています。そこで本論文でもこのMTTRに焦点を絞って検証していくことにします。

またこの可用性を妨げる要因も大きく二つのカテゴリーに分類できます。それはプランされた停止時間（すなわち計画停止）と、プラン外の停止時間（すなわち障害）です。プランされた停止時間にはハードウェアやソフトウェアを含めた製品のバージョン・アップやアプリケーションの変更、データベースのバックアップおよびデータ再編成（Reorg）などのメンテナンス時間などが含まれます。これに対しプラン外の停止時間にはシステム障害（CPU、メモリー、パワー・サプライなど）、ソフトウェア障害（OS、データベース、ミドルウェア、アプリケーションなど）、メディア障害（ディスク障害、データ不整合など）、さらにはユーザー・エラーなどが一因となることもあり、決して単純ではありません。本来であればこれらのすべてに対して稼働し続けるシステム設計が求められます。そのために計画停止時間を最小限にとどめるオンライン・バックアップ機能や、システムを停止せずにメンテナンスを行うオンラインReorg機能などがサポートされています。

本論文では、計画停止時間の可用性を高める手法・機能に関しては別の機会に譲ることとし、プラン外の停止、すなわち障害時対応に的を絞ったデータベース・システムの手法について論述します。

3. 24時間ノンストップのデータベース・システム実現へ向けての手法

24時間ノンストップのデータベース・システムを実現するには大きく分けて三つの手法があると思われます。一つ目は障害時にノードを引き継がせる手法、二つ目はログ転送機能を用いてデータベースを2重化させておく手法、そして三つ目がデータベースを複数に分散し、いずれかのノードが障害でダウンしても業務を遂行できるようにしておく手法です。

以下の各節で、それぞれの手法の特徴・考慮点などを説明していきます。

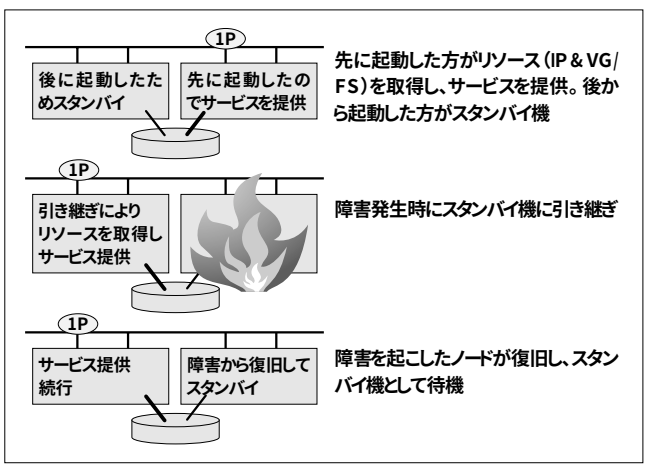

図1．HACMPにおけるホット・スタンバイ構成

3.1. ノード引き継ぎを用いた手法

データベースの障害時対応において最もポピュラーな手法といえるのがHACMP（High Availability Cluster Multiprocessing）を用いて障害時にノードを引き継がせる手法です。障害時対応を考慮したシステムには今や必須構成となっています。構成および機能の概略を図1に示します。

HACMPは障害が起こったノードを検知し、そのノードのリソースを別の待機中のノードに引き継がせ、処理を続行させるといった手法を用います。このとき検知できるのは、CPU障害やネットワーク障害などです。ここで一つ注意しなければならないのは、HACMPはディスク障害には対応できない点です。CPUやネットワークの替えはいくらでも利きますが、データの替えは利きません。待機していたノードが、障害のあったノードのデータを2重持ちしているわけではないので、データだけは引き継がねばならないのです。ディスク障害にはRAIDかミラー化で対応するしかありません。また障害ノードのメモリー上にあった情報も引き継げないため、ノードを引き継いだ後にデータベースを起動させるにはロールバック処理を行う必要があります。

このディスク引き継ぎ（マウント処理）とデータ整合性のためのロールバック処理に時間が取られてしまうために、どんなにチューニングしても引き継ぎには最低でも10分はかかってしまいます。数年前のシステムであればこれで十分かもしれませんが、現在求められている24時間ノンストップのデータベース・システム、特に証券を扱うような特定時間の稼働率に100%を求めるようなシステムには対応しきれません。ただし後述するデータ・レプリケーション機能などと組み合わせることで、より先進的な手法を生み出しているのも事実です（詳細は後述）。そういった意味で、やはりこのノード引き継ぎを用いた手法は依然として障害時対応の基本であり、必須機能と考えて間違いないと思われます。

もう一つ、ノード引き継ぎを用いた障害時対応として仮想サーバーという概念を持ち込んだOracleデータベース（以下、Oracle）のフェイルセーフという手法が挙げられます。これはプライマリー・ノードとセカンダリー・ノードでフェイルセーフ・グループを構成し、それがフェイルオーバーの単位となる仮想サーバーとなります。クライアント・アプリケーションからはどちらのノードと接続するのか考慮する必要はなく、仮想サーバー・アドレス（フェイルセーフ・グループ・ネットワーク・アドレス）に向かってコネクションを行い、処理を依頼します。そして障害がプライマリー・ノードで発生した場合は、待機していたセカンダリー・ノードに引き継がれますが、クライアント側からは仮想サーバー・アドレスへの接続要求は変わらないため、ユーザーは障害を意識することなくアプリケーションへの影響も最小限度にとどめることができます。フェイルセーフの概要を図2に示します。

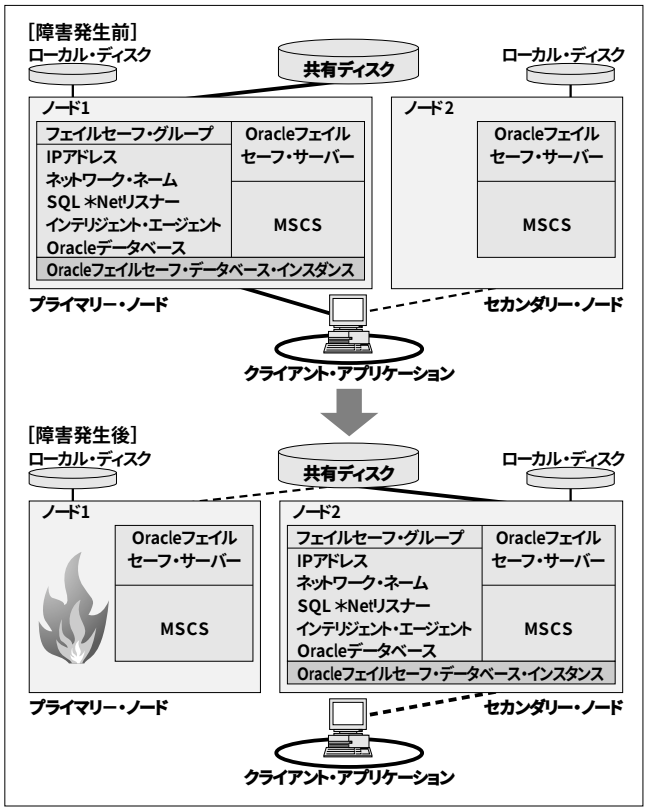

図2. フェイルセーフにおける障害時の動き

フェイルオーバーの主な動きは次のようになります。プライマリー・ノードに障害が発生すると、まずフェイルセーフ・グループに対して指定されたフェイルオーバー方針が実行されます（この方針はユーザーが任意に指定可能）。一般的なフェイルオーバー方針では、プライマリー・ノード側でデータベースの再始動を指定回数だけ試行します。それでも駄目な場合にノード引き継ぎ（フェイルオーバー）が始まります。障害の起きたプライマリー・ノードに接続されていたディスクを、セカンダリー・ノードに再マウントさせます。共有ディスクはプライマリーとセカンダリー両方のクラスター・ノードに同時に接続されていますが（2重パス）、ある時点に特定のディスクを直接マウントしてアクセスできるノードは一つに限定されるため、障害発生後にセカンダリー・ノードにマウントし直す必要があるからです。そしてフェイルセーフ機能はセカンダリー・ノード上でインスタンスを起動し、データベースを回復します。障害のあったプライマリー側のメモリー情報は引き継げないため、コミット処理されていない業務をロールバックさせるのです。ただしフェイルセーフ・グループ単位で管理するインテリジェント・エージェントは一緒にフェイルオーバーされます。このエージェントはデータベースにスケジュールされたジョブやイベントを管理するものであり、引き継がれたセカンダリー・ノードでジョブやイベントが確実に監視され実行さ

れることになります。引き継がれたセカンダリー・ノード上のデータベースの回復時間は、チェックポイントが最後に行われた時期や、REDOログの処理に要する時間、およびこの作業に使えるシステム・リソース（例えばメモリーやCPUサイクルなど）に依存することになります。

この手法は、IBMにおけるHACMPとDB2 UDBの組み合わせによる手法に酷似しますが、残念ながらHACMPのような障害を検知する機能は持ち合わせていません。そのため他ベンダーのクラスター・ソフトウェア（例えばMSCS：Microsoft® Cluster Server）などと組み合わせたソリューションとならざるを得ません。この場合、クラスター・ソフトウェアの役割は障害を検知してディスクを引き継がせるところまでとなります。

3.2. ログ転送によるデータベース2重化を用いた手法

たとえディスクをミラー化し、障害時にノードを引き継がせる構成にしたとしても、何らかの災害でそのサイトもろともダウンしてしまえば元も子もなくなってしまいます。その備えとしてバックアップ・イメージを取得しているわけです。ただ、それを使ってデータをリストアし、障害のあった直前までログを使ったロールフォワード処理を別のマシンで行おうとすると、データ量やバックアップ取得のタイミング周期にもよりますが、かなり時間がかかることは否めません。T（テラ）データ規模のシステムともなれば数日かけても復旧は厳しいかもしれません。こういった観点からリモート・サイトにバックアップ・システムを構築するという発想で生まれたのが災害対策システムであり、それを実現しているのがGeoRM（Geographic Remote Mirror）です（図3）。

もともとは広域災害発生時のデータ保持を目的としたものですが、使い方によってはノンストップ・システムに応用が可能です。この機能の仕組みは、簡単にいえばデータの2重書きです。データの更新時にリモートとローカルの両方のディスクに書き込むことにより、リモート・サイト（セカンダリー・ノード）にもローカル（プライマリー・ノード）と同一のデータを保存することを可能にしたのです。これはほかの手法と比較しても、障害時に整合性の取れたより新しいデータを提供できることや、迅速にデータが使用可能であること、またあらゆるアプリケーションのデータに使用できることなどの点で優れています。ただしあくまでもシステムから見たデータ・ファイルを基にしており、データベースを考慮しているわけではありません。

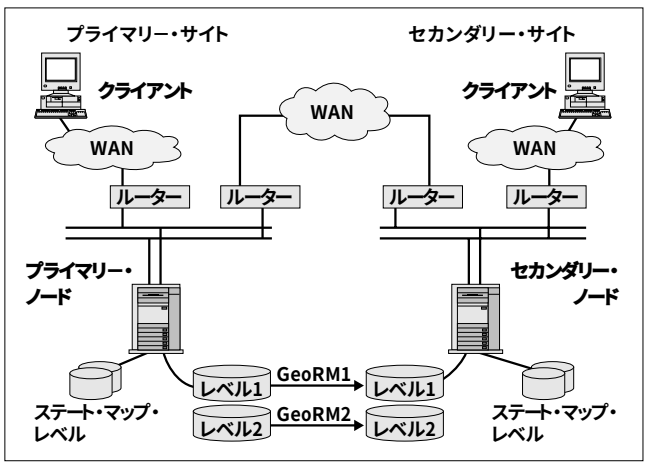

図3. 災害対策システムにおける2サイト構成例

つまりデータベースを丸ごと2重にするにはすべてのデータを対象にする必要がありますが、それではネットワークに非常に負荷がかかります。ただしデータベースとしては更新データはログだけあればよいので、ログ・ファイルのみを2重書きし、それをリモート・サイトで常時ロールフォワードできる体制（ただしこれはGeoRM側ではできないためデータベース側で考慮する必要があります）にしておけば最新のデータベース状態で瞬時に引き継ぐことが可能となります。

考慮事項としては、障害引き継ぎ後のノード間の整合性をどのように合わせるかが挙げられます。障害のあったノードが復旧してきた場合、いずれかの時点でデータの整合性を取る必要があるからです。そのためにリモート・サイトは一つではなく二つ置いて、ローカルと合わせて計3サイトの3重書き体制にしておくのも一つの手法として考えられます。

もう一つの大きな問題点として、GeoRMはサイト障害の監視機能を持っていないことが挙げられます。つまりサイト障害が発生するとリモート・サイトにデータのコピーができなくなるため、結果的にサイト障害が発生していることは分かりますが、通知や自動的な復旧は行いません。GeoRMはデータの整合性を保証するだけなのです。従ってGeoRMとは別の機能によってマシン障害を検知・通知する仕組みをつくる必要があります。

またGeoRMは、HACMP機能との並存を考慮した設計とはなっていません。併存させるにはGeoRMで使用するネットワークやロジカル・ボリュームも、プランニングの段階から通常のネットワークやロジカル・ボリュームをHACMP化する場合と同様に考慮すれば、お互いを補完し合うより強固なノンストップ・システムを構築できます。

そこで登場したのがHAGEO（High Availability Geographic Cluster）です。HAGEOは、GeoRMの持つリモート・サイトへのミラーリング機能とHACMPが協調稼働して、より高い可用性を実現する手法です。基本的な動きとしては、障害がローカル・サイトで検知された場合はHACMP機能で引き継ぎます。もしサイトごとダウンした場合はリモート・サイトで引き継ぎます。うまく応用すると、例えばローカル・サイトのノードがダウンしてHACMPで引き継いでいるときには、クライアントからのアクセスをリモート・サイトで引き継げます。これによりHACMP

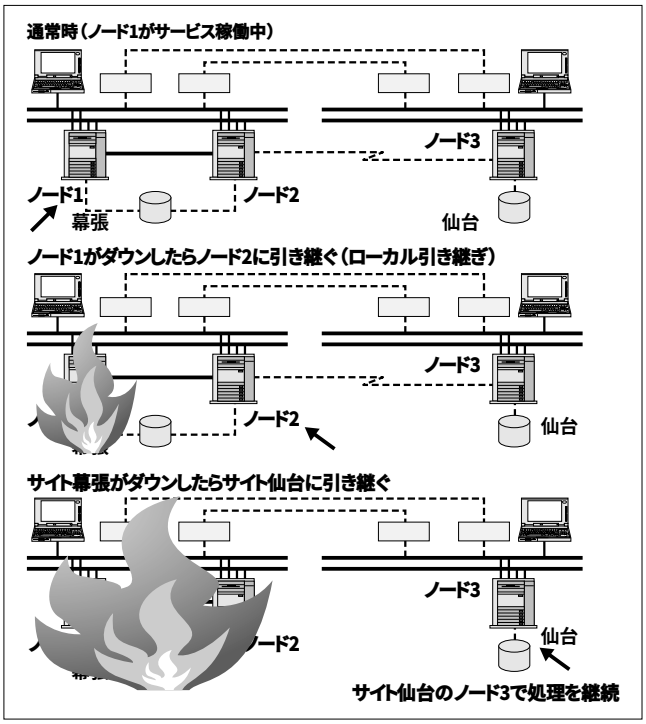

図4．HAGEOによる障害時の動き

での引き継ぎ時間のタイム・ロスを補うことが可能となります（図4）。また障害復旧後のサイト間のデータ整合性の問題においても、HACMPで引き継ぐわずかな時間（10分前後）であれば再度ミラーリングし直す負荷はそれほど掛からないと思われます。

この結果、GeoRMの手法で必要だった3サイトの3重書きという方式は不要となり、ローカルとリモートの2サイト体制で十分ということになります。さらに、HACMPでノードを引き継いでいる間はデータ更新を不可とするようにアプリケーション上のロジックで管理すれば、障害復旧後のサイト間のデータ整合性処理も不要となります。またHAGEOは、HACMPの持つサイトやマシン障害を検知・通知する仕組みを持っているため、GeoRMのように仕組みを別途作成する必要もなくなります。

さらにデータベースにおいてログ転送によるデータ・レプリケーション機能を用いた手法が挙げられます。この手法のコンセプトは、GeoRMやHAGEOによるデータの2重書きに似ていますが処理の流れが多少異なります。まず業務を遂行しているノード（プライマリー・ノード）でデータベースに変更が発生すると、CAPTUREと呼ばれるコンポーネントがログから変更データを収集し、ステージング表と呼ばれるものを自動生成します。それを、待機しているノード（セカンダリー・ノード）のAPPLYと呼ばれるコンポーネントが定期的にチェックし、変更を確認して収集を行い、自分のノードのテーブルにデータを適用することでノード間の整合性を取る仕組みとなっています。GeoRMなどと異なるのは、完全に同期の取られた2重書きの手法ではなく、プライマリー・ノードの変更をトリガーとした非同期型の手法となる点にあります。そのためデータ・コピー中に障害が発生すると、データの整合性が損なわれる恐れもあります。また障害ノードが、回復後に引き継いだノードとのデータの整合性を再調整する仕組みを必要とする点もGeoRMの考慮点と同じです。

この手法のメリットは、データベースが提供する機能であるため、ミドルウェアが提供するファイル・コピーとは異なり、セカンダリー・ノードのデータベースへの変更データ（差分データ）の適用は自動で行われ、そのための仕組みづくりを別途考慮する必要がないことです。つまりセカンダリー・ノードのデータベースは常にプライマリー・ノードと同期の取られた状態に保つことが可能となります。

しかしこの機能はもともと可用性のためのものではなく、この機能を用いただけではノンストップ・システムの構築が不完全であることは否めません。そこで前述したHACMPと組み合わせることで、高可用性システムを実際に構築している事例を紹介します（図5）。この手法はHAGEOの手法とかなり似ています。すなわち業務を遂行しているプライマリー・ノードで障害があった場合、データ・レプリケーション機能を用いて同期を取りつつ、待機しているセカンダリー・ノードにクライアントからの接続を素早く切り替えます。その間にプライマリー・ノードでは、

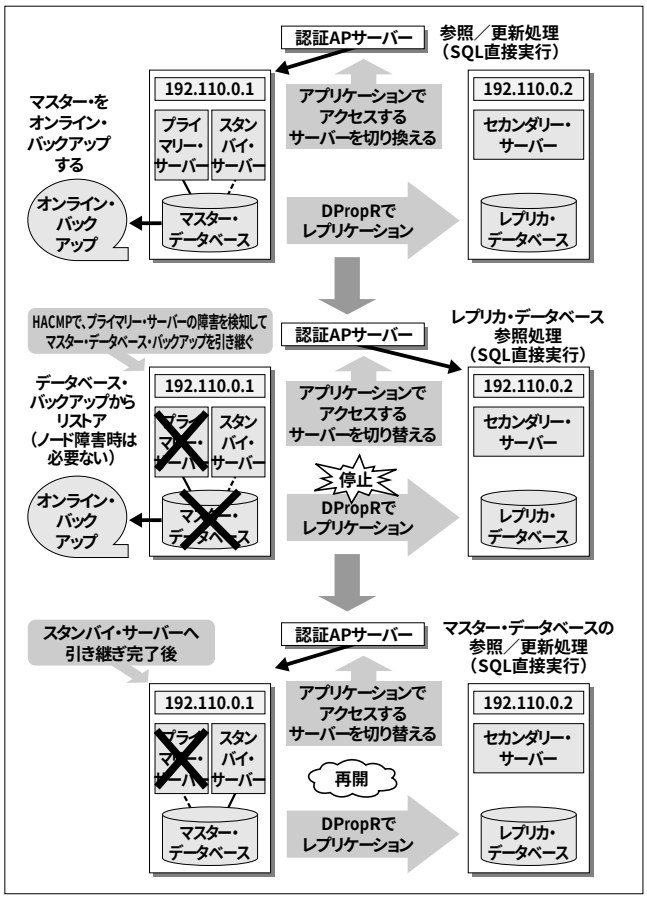

図5．HACMPとデータ・レプリケーションを組み合わせた手法の障害時の流れ

HACMP機能を用いて障害ノードからテークオーバーを行います。HACMPの引き継ぎが完了すると、セカンダリー・ノードへの接続を、再びプライマリー・ノードへ切り替えて業務を続行する仕組みです。このシステムにおける唯一の考慮点は、セカンダリー・ノードに引き継がせている間（HACMPで引き継いでいる間）に起こった更新データの整合性を再びプライマリー・ノードと取る仕組みづくりにあります。本事例では、その仕組みづくりをスキップするためにセカンダリー・ノードに引き継がせている間は更新処理を行わないアプリケーション・ロジックで対応しています。この制限がクリアされればかなり完成度の高いノンストップ・システムに近づくものと思われます。

DB2 UDBには、ESS（Enterprise Storage Server™）が提供するフラッシュ・コピーや、OSが提供するファイル・コピーを基にしたリカバリー手法は今までサポートされていませんでした。つまり障害からの回復手法は、データベースが提供するバックアップ／リカバリー方法しかサポートされていなかったのです。そのためバックアップはオンライン時に取得したとしても、障害が発生してから回復するまでリストアやロールフォワードに膨大な時間がかかり、その間は業務を停止せざるを得ませんでした。しかしDB2 UDB V7.2からはOnline Split Mirrorという機能がサポートされました。これにより、オンライン状態のデータベースからフラッシュ・コピーのような機能を用いてデータベースのミラー・イメージを瞬時に取得することが可能です。これはデータベースを一時的に書き込み禁止、すなわち読み取り専用の状態にします。その間にフラッシュ・コピーを使ってデータの整合性の取れたデータベースのミラー・イメージを別のセカンダリー・ノードのESSディスクなどに取得します。その後、セカンダリー・ノード上でデータベースをロールフォワード・ペンディング状態のまま待機させるのです。プライマリー・ノードでの更新処理は、アーカイブ・ログとしてUSEREXITプログラムなどを通してセカンダリー・ノードに送られ、そのたびにロールフォワードが実行されてプライマリー・ノードと同期を取る仕組みとなります。もしプライマリー・ノードに障害が発生したときはセカンダリー・ノードのロールフォワードをコンプリートにし、セカンダリー・ノード側のデータベースを使用可能にします。

この手法の考慮点は、プライマリーとセカンダリーとの整合性はアーカイブ・ログ・レベルであるという点です。つまりプライマリー側にあるアクティブ・ログの情報はセカンダリー側には反映されないのです。これを回避するために、前述したGeoRMと組み合わせた手法が考えられます。つまりUSEREXITでログを飛ばす代わりに、GeoRMを使ってログ・ファイルを同期化します。この手法だとアクティブ・ログ・レベルまでの同期化が可能となりますし、USEREXITでのログ・ファイル転送処理によるプライマリーとセカンダリーのデータ不整合の時間のずれを解消することが可能となります。ただしこの手法には、障害回復後のプライマリーとセカンダリーのデータの再同期といった問題が残ります。これは現時点では、オンライン中のセカンダリーからのフラッシュ・コピーでデータベース・イメージをプライマリー側にミラー化し、ロールフォワード・ペンディングで待機するデータベースを作成し直すしか、手だてはないように思われます。

Oracleには前述したフェイルセーフとは別に、ログ転送による障害時対応に特化した手法があります。スタンバイ・データベース手法です。この手法は上記のフェイルセーフではディスク障害には十分に対応できない部分をノード切り替えによって解決しようという発想です。そのためデータの2重持ちの必要がありますが、その手法は、IBMにおけるGeoRMとDB2 UDBにおけるスプリット・ミラーを組み合わせたようなものとなります。つまりプライマリー・ノード側のオンラインREDOログがアーカイブさせるタイミングで、セカンダリー・ノードにそのアーカイブREDOログを自動転送し、セカンダリー・ノードに用意されているスタンバイ・データベースに随時適用し、プライマリー側と同期を取る仕組みです（図6）。

この手法の優れているところは、すべての流れが一貫してOracleの機能として働くため、アーカイブREDOログの転送からスタンバイ・データベースへのログの適用まで、すべて自動で行われる点です。IBMの手法では担当する製品が分かれてしまうため、データの転送や転送後のログ・データの適用はマニュアル作業となってしまい、何かしらの仕組みのつくり込みが必要となります。ただしGeoRMを使った場合はスタンバイ・データベースと異なり、ログ・データの転送ではなく完全2重書き手法なので、転送部分のつくり込みは不要となります。またGeoRMを使った場合はアーカイブされる前のアクティブ・ログ（Oracleでいう

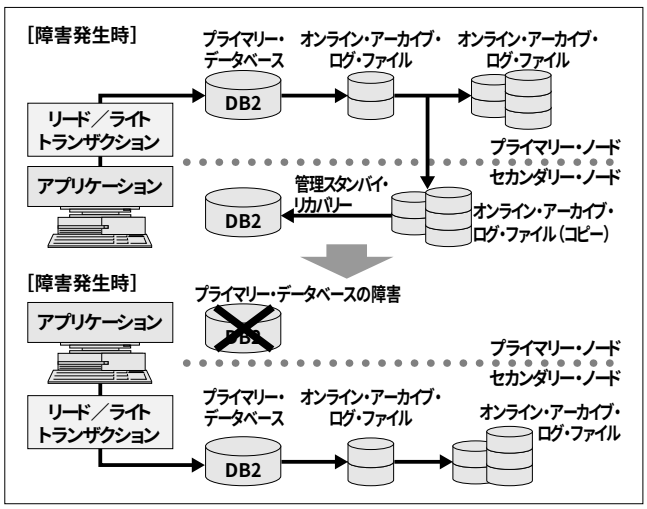

図6．スタンバイ・データベースによる障害時対応

オンラインREDOログ）の適応まで考慮することができますが、スタンバイ・データベースの手法ではそこまでの整合性は保証されません。またログ・データの転送中の障害時対応や、ログ転送のために起こるプライマリー・ノードとセカンダリー・ノード間でデータの整合性を取るまでの時間のずれが大きいといった問題も残されます。さらにOracle社にはIBMにおけるHAGEOのような手法が存在しないため、障害復旧後のプライマリー・ノードとセカンダリー・ノードとのデータ整合性をどう取るのかといった問題も残ります。この点については、新しいアーキテクチャーを踏まえながら次章で検証していくことにします。

3.3. データベースの複数分散化を用いた手法

三つ目の手法としてデータベースの複数分散化、すなわち並列データベースを用いた手法を取り上げます。

データベースを並列に分散化させることで危険を分散させ、いずれかのノードがダウンしたときにもほかのノードで処理を続行させるというのが、並列構成による高可用性手法のもともとの発想です。例えばe-ビジネスにおけるインターネット・システムのWebサーバーなどがよく用いる手法です。ただしここで注意したいのは、複数台のWebサーバーなどを立ち上げ、処理を分散化させ、いずれかのサーバーがダウンしても処理を続行できるのは、並列で稼働している複数のWebサーバーがそれぞれ所有するデータはセッション・データベースに一元管理されている点です。それ故にいずれかのWebサーバーがダウンしたとしても、別のWebサーバーがデータを引き継いで処理を続行できるわけです。

これと同じことをデータベース側にも求めようとすると、複数のデータベースのノードから、データベース・データも一元管理される必要があります。すなわちシェアード・ディスク方式にする必要があるのです。ところがIBMのUNIX系並列データベース（DB2 UDB/Extended Enterprise Edition。以下、EEE）は残念ながらシェアード・ナッシングというアーキテクチャーを採用しています。これはCPUもメモリーもそしてディスクも、ノード間で一切シェアしないという方式です。この方式の利点はパフォーマンスとスケーラビリティーにあります。つまりノードを並列にどんなに増やしていってもボトルネックになる部分がないため、理論的にはリニアにパフォーマンスが向上していくことです（実際はノード間でのデータ転送などの負荷があるため完全なリニアにはなりませんが）。これにより理想的なスケーラビリティーが望めることになります。一方、データの一元管理を行うシェアード・ディスク方式では、ノードが増えるほどシェアしているディスクへのI/Oが集中するためにボトルネックとなってしまい、パフォーマンスとスケーラビリティーが思うように上がりません。

EEEを導入すれば、Webサーバーのように可用性が高まるという考えは間違いありません。シェアード・ナッシング方式のEEEではディスクはノード間でシェアせずに、各ノードは各自のデータを管理しているため、いずれかのノードが障害でダウンした場合にほかのノードがデータを引き継ぐことはできないのです。

この問題を解決するには、今のところHACMP構成を用いる以外に手だてはありません。高可用性のために並列データベースを用いるのならシェアード・ディスク方式を選択する必要があるのです。

ここでシェアード・ディスク方式を採用しているOracleデータベースの並列手法を検証したいと思います。Oracle社における並列データベースは以前はOracle Parallel Server（以下、OPS）と呼ばれていましたが、現在はRAC（Real Application Clusters）という名前に変わっています。その位置付けはIBMのDB2 UDBにおけるEEEと同じですが、もともとの発想や役割は根本から異なっています。図7に並列データベースのアーキテクチャーの比較を示します。

並列データベースを持つUNIX系ベンダーの中で、このアーキテクチャーを採用しているのはOracle社だけです。並列データベースが脚光を浴びた時期には、このOPSは非常に不利な対場に立たされていました。当初、並列データベースの主な用途はDSS業務（意思決定支援システム）でした。つまりデータ・ウェアハウスのような膨大なデータから複雑なSQLを駆使してどれだけ高速にデータを取得できるかがカギだったのです。つまり検索系の処理がメインでした。こうした処理には、データを一元管理するシェアード・ディスク方式は苦戦を強いられました。どんなにノードを並列に増強（つまりCPUやメモリーを並列化）したとしても、ディスクへのI/Oが一点に集中することでボトルネックが発生し、思うようなスケーラビリティーやパフォーマンスが望めなかったからです。この状況は、e-ビジネスにおけるインター

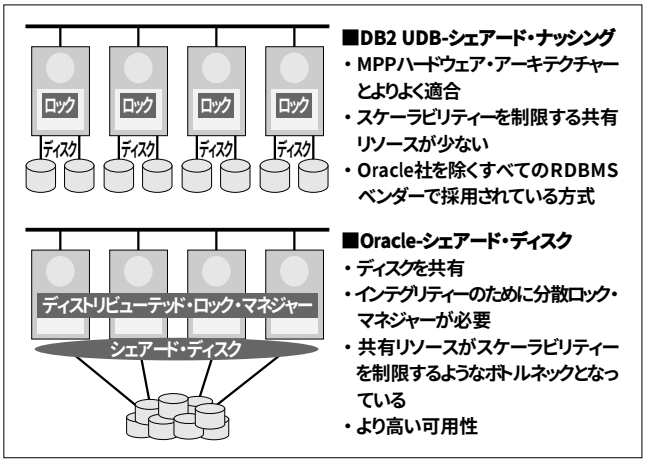

図7．並列データベースにおけるアーキテクチャーの比較

ネット・アプリケーション時代が到来して一変しました。検索だけではなく、更新系の処理（Webからの購買や発注など）が不可欠な環境になったからです。DSS主体の時代のように、ホスト系のシステムからデータを夜間バッチで取得し、それを日中にオンライン検索していたときとは異なり、更新されたデータが並列データベースだけに存在するようになりました。こうなると求められる信頼性は格段と異なり、先にも述べたように業務の24時間化に伴う高可用性がパフォーマンスやスケーラビリティーにも増して重要になってきました。こうした環境では、シェアード・ディスク方式は大変有利です。ノード障害に対する可用性には非常にマッチしたアーキテクチャーだからです。いずれかのノードがダウンしたとしても、シェアード・ナッシング方式とは異なり、データベースのデータは全ノードからシェアされているためにほかのノードがデータ・処理を引き継ぐことができます。

このときメモリーの情報まではシェアできないため、障害のあったノード上のメモリー情報は処理がコミットされていなければロールバックされることになります。とはいえアプリケーションの特別な変更は一切不要であり、この並列データベース方式を採用するだけで24時間ノンストップ・システムの実現に限りなく近づくことになります。この方式の課題点はパフォーマンスとスケーラビリティーでしょう。e-ビジネス環境では高可用性と合わせてシステムの拡張性も強く求められます。この相反する要件を解決することができれば究極のノンストップ・システムの実現にまた一歩近づくでしょう。この点に関しては、Oracle社の考える新しいアーキテクチャーなどを交えながら、次章で考察していくことにします。

4. 今後の24時間ノンストップ・データベース・システムへの取り組みと展望

前章までで24時間ノンストップのデータベース・システムを構築するための手法を述べてきました。e-ビジネスにおける24時間稼働のアプリケーションが主流となりつつある今日、データベース・システムの高い可用性が必要とされることに異論はないでしょう。

また、それに関連してe-ビジネス環境でのシステムに求められる強い要件として、柔軟なスケーラビリティーが挙げられます。ビジネスの拡大に伴って要求される処理能力の増強に、システム（特にデータベース・システム）は柔軟に対応できなければなりません。インターネットを用いたe-ビジネス・システムは、処理の負荷を想定するのが非常に困難です。従来のイントラネット・システムであれば、最大同時アクセス・ユーザー数などを社内の人数・端末数などから想定するのはそれほど難しいことではありません。ところが相手がインターネットとなると、どれだけのユーザーがどれくらいアクセスしてくるかは、やってみなければ分かりません。また社内システムならば、処理のピーク時期の予測やサービス提供時間などの管理も可能でしたが、e-ビジネスではそうもいきません。例えばゲーム・ソフトウェアの販売などでは、今まで日に数百～数千のアクセスだったものが、一つのヒット商品が出ることでいきなり数百万単位のアクセスに膨れ上がることも珍しくはありません。また前にも紹介した証券のシステムなどでも、株の動きなどによってアクセス数は大きく変動しますし、それがいつ、どれくらいの規模になるのかを予測するのは非常に困難です。こうなると企業は、システムを柔軟に拡張／縮小したいと考えます。どの程度使われるのか分からないシステム（特にデータベース・システム）のために投資する余裕は今日の企業にはありません。

しかしデータベースは、ほかのWebサーバーなどのように簡単にスクラッチ／ビルドができるものではありません。Tバイト級に膨れ上がったデータベースともなると、処理速度の高い新しいサーバー・マシンが発表されたからといっても、そう簡単に移行できるものではありません。SMPマシンであったとしても、そのマシンがサポートしているCPU数やメモリー・サイズが拡張の限界となってしまいます。

そうなると、柔軟な拡張性と無限に近いスケーラビリティーを備えた並列データベースが重要な位置を占めることになります。また高可用性とも関連しますが、拡張に伴うシステム停止時間も最小限にしたいと、企業は考えます。しかも、業務に影響を与えないように、システムを止めることなく拡張できるシームレスなサポート環境を望んでいます。

GeoRMやHAGEO、スプリット・ミラー、スタンバイ・データベースなどを用いたログ転送によるデータベースの2重化手法は、高可用性を念頭に置いた手法です。この手法の特徴は初めからデータ障害を意識し、データの引き継ぎを考慮している点です。そのためにどうやってデータを2重持ちさせるのか、それをどうやってデータベースに適応させるのか、またユーザーに障害を気付かれずにセカンダリー側のシステムに引き継がせることに焦点を当てています。それぞれの仕組みが異なるとはいえ、結局やろうとしていることは同じです。どの仕組みでも問題となるのは、障害復旧後の切り替え処理（プライマリーとセカンダリーのデータの整合性合わせ）です。セカンダリーが待機しているときは瞬時に切り替えられたとしても、プライマリーが障害復旧をしている間にセカンダリー側で更新処理が多発してしまうとプライマリー側が復旧したときのデータの整合性に大きな食い違いが生じてしまいます。

このデータの整合性処理を最小限度に抑えるために、IBMではHACMP機能を絡ませたHAGEOとDB2 UDBのスプリット・ミラーなどを考案しています。またHACMPとデータ・レプリケーションを用いてこの問題をクリアしようという試みも紹介しました（ただしHACMPでプライマリー側のノードを切り替え中にはセカンダリーへの更新はアプリケーション・ロジックで回避しています）。このような機能はいまだOracle社などの他社製ベンダーではサポートしていません。これはIBMの大きな強みといってよいでしょう。

ただしOracle社はその代わりにOracle Data Guardなる新しいアーキテクチャーを持ち込んでいます。これにはプライマリーとセカンダリー（スタンバイ）との間で役割を交互に切り替えること（スイッチオーバー／スイッチバック）を可能にしようとするものです。これにより障害復旧後のデータ整合性を取る仕組みをデータベース機能だけで実現し、ユーザーなどのつくり込みは不要となります。またこの機能は、計画停止のときなどにも威力を発揮するものと思われます。さらにこのアーキテクチャーではセカンダリー・サイトへのログ転送サービスも強化されています。ログは同時2重書き方式ではなく、相変わらずデータ転送を用いてはいますが、今までアーカイブREDOログ単位での転送であったものが、プライマリー側でのオンラインREDOログへの書き込みに同期してセカンダリー（スタンバイ）側へオンラインREDOログ情報を転送するものに変更されています。これにより障害発生時のデータ損失を一段と抑えることが可能となりました。

またもう一つの弱点であったネットワーク障害などによるアーカイブREDOログ転送漏れ処理に関しても、アーカイブREDOログが到着するまではオンラインREDOログ適用を遅延させるというデータベースの自動回復機能が盛り込まれています。このようにOracle社は着実に自分の手法の弱点を補いつつ、高可用性手法に磨きをかけてきています。

ここで考慮したいのは、Oracle社などのような専門ベンダーはすべての手法をデータベース機能として一元化し、でき得る限りシームレスな自動化の方向を目指している点にあります。それに対しIBMは、データベースの機能だけでなく、ミドルウェアなどと共同したチームワーク体制で臨もうとしています。それに伴い、お互いの機能をつなぎ合わせたり補完し合う部分はつくり込みにより対応ということになります。

もう一つの有望なものとしてデータベースの分散化を用いた手法が挙げられます。これはノード障害に特化した手法です。この手法の特徴は、データを一元管理しておいて、いずれかのノードがダウンしたときにほかのノードで処理を続行させようという発想です。

IBMの並列データベースであるEEEは、残念ながらシェアード・ディスク・アーキテクチャーを採用していないため、高可用性システムの解法には向いていないことは既に説明しましたが、どちらかというとe-ビジネスのもう一方の要件である拡張性やスケーラビリティーに優れた手法といえます。それに対しOracle社の並列データベースは、この高可用性システムに最も有望な手法の一つといっても過言ではないでしょう。ただこの手法の課題点は拡張性とスケーラビリティーにあります。この問題をクリアできなければ今後のe-ビジネスの世界で勝ち進むのは困難です。

その点はOracle社もOPSに新たなアーキテクチャーを搭載し、RACとして大きく進化させて対応しようとしています。新しいアーキテクチャーの目玉は何といってもキャッシュ・フュージョン機能と思われます。並列データベースとして分散されている複数のノード間のメモリーを、グローバルに一元管理しようというものです。つまり、ある検索処理がいずれかのノードに依頼された場合、まずそのノード上のメモリー（バッファー・キャッシュ）の中を探します。今までは、もしメモリー上に見つからなければディスクからデータを持ってこようとしました。ところがキャッシュ・フュージョン機能では、ディスクにアクセスしに行く前にほかのノードのバッファー・キャッシュ・メモリー上に検索依頼されたデータがないかを問い合わせます。もしほかのノードのメモリー上に見つかればそれをノード間転送で受け渡します。これによりディスク・アクセスへの一点集中によるボトルネックというシェアード・ディスクの弱点を緩和しようというもくろみです。これは更新処理がバッティングした場合にも有効だといわれますが、さすがに1件ずつコミットするような同時アクセス更新処理の対応には、やはり処理がシリアライズされてしまうのは致し方ないところでしょう（もしメモリーの中だけで更新し合ってディスクに書き出さないとすれば、それこそ障害時にデータの整合性の観点から大変なことになりかねません）。

ただしここで重要なのは、Oracle社などの他社ベンダーは、自分の手法の持つ相反する特性を考慮しながら、時代の要請に合わせてその弱点を補強するような努力を果敢に行ってきていることです。

5. おわりに

e-ビジネス時代のアプリケーションは、無数の企業を対象とし24時間ノンストップで稼働することが半ば宿命的に要求されます。そのような環境におけるデータベース・システムの障害時対応は、Webサーバーのように並列に複数を立ち上げて処理させられないだけに、その重要性はますます増しています。

このためIBMをはじめとするベンダーはITの粋を駆使し、一歩でも理想に近いノンストップ・システムの実現に向けてしのぎを削っています。ただデータベース専門のベンダーがすべて自前のデータベース機能で解決させようとしているのと対照的に、IBMはハードウェア（ディスク装置）やミドルウェア、データベースのそれぞれの持つ機能・特徴を組み合わせ、総結集して対応しようとしているように思えます。これはある意味では何でもそろい何でもできるIBMの強みともいえるでしょう。

しかしお客様からは見れば、あくまでも欲しいのは個々の機能やアーキテクチャーなどではなく、24時間ノンストップで稼働できるソリューションなのです。時代の要請にIBMのチームワーク力が問われているといっても過言ではありません。本論文がそのようなe-ビジネス環境における24時間ノンストップのデータベース・システムを実現しようとしている方々に、少しでもお役に立てば筆者の大きな喜びです。

（ページ数および表記上の観点から、著者の了解を得て編集部にて手を入れてあります）

[参考文献]

[1] Laurence Clarke, "Highly Available E-Business Solutions with Oracle Fail Safe", Oracle Corporation, 2000

[2] Lance Ashdown,Anna Logan, "Oracle8i Stanby Database Concepts and Administration, Release2（8.1.6）", Oracle Corporation, 2000

[3] Mark Bauer, "Oracle8i Parallel Server Concepts, Release2（ 8.1.6） ", Oracle Corporation, 1999

[4]『Oracle Parallel Server』オラクル・テクニカル・ホワイトペーパー、1998年

[5] Barry Mellish, "Implementing ESS Copy Services on UNIX and Windows NT／2000", IBM Redbooks, 2001

[6] Chris Owen, "eServer pSeries Data Backup Techniques", IBM Corporation, 2001

[7] 奥井 敦子「e-businessにおける大規模DB構築への挑戦」2000年、IBMプロフェッショナル論文

[8] HACMP/6000 Customization Examples, IBM Redbooks, SG24-4498-00, 1995

[9] 宮下 亜津子「GeoRM機能と構成」SIL、2000年

[10] 近本 達士「HAGEO概要とDB2を用いた構成例」SIL、1997年