# データベース・サーバーにおける連続稼働実現への考察

## Study of realization of continuous operation in database servers

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
シニアITスペシャリスト
**房 律子**

**Ritsuko Boh**
Senior IT Specialist
High Availability Systems
System Center No.2
IBM Japan Systems Engineering Co.,Ltd.

**インターネットを使用したe-ビジネス・アプリケーションなどが普及する中で、システムの安定稼働および連続稼働という高可用性への取り組みが注目を浴びています。**
**そこで本論文では特にデータベース・サーバーに関して、高可用性のための必要条件を具体的に提唱します。もちろん連続稼働については、各システム・コンポーネントを網羅して考慮すべきですが、データベース・サーバーは連続稼働を妨げるSPOF（Single Point of Failure）となる場合が多いため特に取り上げることにしました。**
**まずデータベースの連続稼働における計画外／計画停止の要因を検討し、現在のハードウェア機能、DBMSとしての機能、データベース／アプリケーション設計上での対応策および改善方法をまとめています。また、今後必要となるであろう機能に関しても言及しています。**
**なお、本論文におけるDBMSとはDB2® UDB FOR OS/390®の機能を中心に述べ、一部DB2 UDB FOR AIX®などの機能も織り込んでいます。**

With the ongoing diffusion of e-business applications using the Internet, more and more attention is being directed toward tackling the challenges presented by the high use potential inherent in the stable operation and continuous operation of systems.
In this paper I offer some concrete proposals for the conditions required to realize this high use potential in connection especially with database servers. It goes without saying that we need to consider the whole range of system components in connection with continuous operation, but I have decided to focus on database servers since these are often the "single point of failure" (SPOF) that stands in the way of continuous operation.
I begin by studying the factors that lie outside the scope of planning and contribute to interruption of a plan in the continuous operation of databases, and I then take a look at responses and methods of improvement connected with the design of current hardware functions, functions as DBMS, databases and applications. I then allude to the functions that are likely to prove necessary in the future.
The DBMS referred to in this paper involves especially the functions of DB2® UDB FOR OS/390®, although I have also partially incorporated the functions of DB2 UDB FOR AIX®, etc.

## 1. はじめに

インターネットを使用したe-ビジネス・アプリケーションに代表されるように、不特定の時間に不特定多数のユーザーにサービスを提供するシステムが要求されています。これらのサービスはその会社の顔であり、何らかの障害によりシステムが停止したり誤動作を起こすと、その会社の評判や業績に対して大きな影響を与えます。既にコンピューター・システムはライフ・ラインの一つであり、不安定なシステムは社会不安さえ引き起こす可能性があります。

こうした環境の変化に伴い、システムの安定稼働および連続稼働という高可用性への要件は、業務要件の単なる延長線上ではなく、システム構築の当初からの要件となってきています。すなわち24時間連続稼働という目的を掲げて業務を設計して、システム構成を判断し、どのように運用設計するかを考慮する必要があるのです。しかし、これほどまでに連続稼働が注目を浴びている今日にあっても、実際にどのような要件を満たせばそのシステムが連続稼働を実現しているかを判断できる情報は意外にまとまっていません。そう感じるのは筆者だけでしょうか。「連続稼働」という言葉だけが独り歩きをしてしまい、それがどのような状態を指し、どのような要件を満たせば実現されるかを標準化し、まとめた資料が少ないのです。

よって本論文では、特にデータベース・サーバーに関して、高可用性のための必要条件を具体的に提唱します。もちろん高可用性システムの構築に当たっては、アプリケーションやネットワークなどすべてのコンポーネントにおけるコンセンサスが必要です。しかし、ここで特にデータベースに関して論じるのは以下の理由によるものです。

例えば、ミッション・クリティカルな大規模システムの構築を考えたとき、従来のIMS™またはCICS®-DB2®といったシステムのほかに、クライアント側はブラウザー機能を持つクライアント・システムを配置し、アプリケーション・サーバーとしてのHTTPサーバー、そしてバックエンドとしてのデータベース・サーバーといったような3層型システム形態をイメージすることが可能です。この3層型システム形態を可用性という観点から眺めた場合、クライアントまたはアプリケーション・サーバーに関しては複数のサーバーを稼働させれば、たとえ1台に障害が発生したとしても、ほかの生きているサーバーが処理を続けることで可用性が向上します。なぜなら、アプリケーション・サーバー同士のお互いの連携処理はなく機能的に独立しているため、理論上その機能はいくらでも複製可能だからです。

ところがバックエンドにあるデータベース・サーバーに多量の更新業務が存在するときには、データのミラーリングなどは非常に困難であり、複数サーバーではなくデータを1カ所に統合化することが必要となってきます。つまり、データベース・サーバーの複製化は極めて困難であり、単一システムに集中することでシステム全体にとってデータベース・サーバーの存在自体がSPOF（Single Point of Failure）となります。そのため、データベース・サーバー上で連続稼働をいかに実現するかが、そのシステムの高可用性のカギであるといって過言ではありません。

本論文では、データベースの連続稼働における計画外／計画停止の要因を検討し、ハードウェア機能、DBMSとしての機能、データベース設計およびアプリケーション設計上での対応策および改善方法をまとめています。また、今後拡張されるべき機能の要件についても言及しています。

なお、本論文におけるDBMSとはDB2 FOR OS/390®（以下、DB2/390）の機能を中心に述べ、一部DB2 UDB FOR AIX®などの機能も織り込んでいます。

## 2. 業務上の形態における連続稼働

連続稼働の要件は、実際には業務形態によって変わります。

### 2.1. 非同期更新業務形態

業務が非同期の更新処理の場合、実は連続稼働はそれほど難しい問題ではありません。例えば3層型システムでMQにより非同期にデータのやり取りが可能な場合は、更新データはいったん分散系サーバー側のデータベースまたはMQに格納され、非同期でホスト・サーバー側に集計されます。

こうした非同期形態であれば、ホスト側が何らかのデータベース障害またはメンテナンスで停止しなければならない場合、停止可能時間は分散サーバー側でデータを保持できる時間であり、許容時間はかなり大きくなります。逆に分散サーバー側の障害は、複数構成により部分停止のみが発生します。つまり、クライアント側での全面障害検知の確率はかなり低くなります。このような非同期業務としては、例えば信販業務や製造関連の発注業務などが考えられます。

### 2.2. 同期更新業務形態

ところが非同期処理が許されない業務形態では、先に述べたようにデータベース・サーバーの停止がクライアント側業務の停止に直接影響するため、可用性に対する要件が非常に厳しくなってきます。それらの要件について順次論述します。

## 3. サーバー本体の高可用性

サーバー自体の堅牢性、すなわちハードウェア上のCPU障害やOS自体のソフトウェア障害による計画外停止、およびシステム保守による計画停止が全面停止をもたらさないようにするには複数構成のシステムを持つことが必須です。

1台のCPU筐体でシステムを構成していれば、そのハードウェアが停止したときに業務が全面停止するのは明白です。そこで、複数のCPU筐体で一つのシステムを構築することが必要となります。現在その最も完成された機能が、S/390®によるシスプレックス環境です。いったんシスプレックス環境へ移行すれば、データ共用機能により複数のCPU筐体上に複数のDB2が存在可能であり、ハードウェア上のCPU障害やソフトウェア上のOS障害により一部が停止したとしても、全面停止する確率は限りなく小さくなります。このシスプレックス環境での障害対応の考え方はN＋1型といわれるもので、例えば通常時に4CECのシスプレックス環境でおのおの75%の稼働率で業務を行っていた場合、1台のCECが障害でダウンしたときには、そのCEC上で行っていた業務を縮退することなくほかのCECが補います。よって、障害対応を考慮する場合、シスプレックス環境を構築し、かつCEC数は2より大きいことが望まれます。これに対しRS/6000®などのサーバー系は、HACMP環境と呼ばれるホット・スタンドバイ型の障害対応を行っています。ただし、CPU障害時にシステムの立ち上げ直しが必要なため、業務の全面停止が発生します。

シスプレックス環境においては、障害要素としてCEC障害のほかにCF障害要素が増加します。非データ共用環境およびデータ共用環境で、それぞれのハードウェアに障害が発生した場合に、DB2における業務への影響をまとめたものが表1です。

表に示したように、シスプレックス環境においても、部分的な停止状態は発生します。例えば、シスプレックス内の一つのCPU筐体すなわちCECが障害で落ちた場合、あるいはサブシステム障害によって一つのDB2メンバーが停止した場合、そのDB2メンバーにより更新中の資源に関してリテインド・ロックが発生し、一時的にほかの生きているDB2メンバーからその資源への参照・更新がエラーとなります。このリテインド・ロックは障害発生によりストップしたDB2メンバーをリスタートすることでリリースされますが、そのためにはOS/390上のARM機能により再スタートを自動的に素早く行うことが望まれます。特に最近ではCICSからMQやDB2を2フェーズ・コミットにより更新したり、Webクライアント側からAIX上のMQとDB2を同時更新するなどの要件もあり、2フェーズ・コミット環境が増加しています。そのために、単にリテインド・ロックを速やかに解決するだけではなく、関連したサブシステム同士をいかに素早くスタートするかが重要となります。今後は、IMSのFRB機能のようにDB2メンバーの再始動を行わなくてもリテインド・ロックを解消できるような機能要求が必要でしょう。

GBP側は、V5より拡張されたGBPの2重化を使用すれば、CF障害時に自動的に回復することが可能になりました。さらにz-OS次期バージョンでは、CF自体の2重化機能が可能になります。これによりDB2では、ロック、SCAのストラクチャーも2重化が可能となり、より完成された形で、CPU障害やOS障害によるシステム全面停止の回避が可能となります。さらに、シスプレックス環境では全体停止をすることなくハードウェア／ソフトウェア・メンテナンスも可能な構成であり、高可用性を要求されるシステム構築のデータベース・サーバーとしては必須の機能と考えられます。

## 4. データベース回復に関する考慮点

データベースにとって最も重大な障害はデータベースそのものが壊れることです。DASD障害などにより物理的に破壊される場合もありますが、実際には間違った更新などによる論理的な障害も発生します。つまり、どんなにハードウェアの信頼性が上がったとしても、データベース回復は必ず考慮しなくてはなりません。しかし、今日では表の大きさが巨大化して時に一つの大きさが数百Gバイトを超えることもあります。さらにRAID5のディスクの普及により、ディスク障害数は確かに減っていますが、逆に極めてまれにランク単位や筐体障害が発生

表1.非データ共用環境とデータ共用環境におけるハードウェア障害時の業務への影響

| ハードウェア障害 | 非データ共用環境 | 評価 | データ共用環境 | 評価 | 備考 |
|---|---|---|---|---|---|
| CPU障害 | 全面停止 | × | 1DB2のみ停止。ARMにより他CECでスタート | ○ | 障害DB2が再始動するまでリテインド・ロックが保持され、一部業務が停止する可能性あり |
| CF障害<br>ロック／SCA | N/A | N/A | 一部の業務が待機 | ◎ | ロック／SCA側のCF障害時は再ビルドが終了するまでのみ数十秒待機 |
| CF障害<br>GBP（非2重化） | N/A | N/A | 一部業務停止が発生 | △ | GBP側CF障害時GBPが2重化されていないとGRECP回復処理が必要なため、数十分業務停止 |
| CF障害<br>GBP（2重化） | N/A | N/A | 一部の業務がスローダウン | ◎ | 2重化されていると切り替え時に若干スローダウンが発生するが業務停止は発生せず。 |

して多くのデータベースが一度にアクセス不能となり、業務のほとんどが停止することを想定しなければなりません。つまり大容量のデータベースを一度に、多量に、しかも可能な限り高速に回復することが要求されているのです。

その実現には以下の方法が考えられます。

- データベースの回復自体の高速化

データベース回復にかかる時間は、バックアップ・ファイルのリストア時間とログを適用させる時間の和であり、それらの時間を短くするための機能が必要です。

- データベースの2重化を行う

障害発生時にデータベース回復を行わず、障害を起こしていない側のデータベースに切り替えることで業務を継続します。

### 4.1. データベース回復自体の高速化

データベース回復を高速に行うために、以下の要件をいかに実現するかを検討します。

(1)障害時にバックアップ・ファイルのリストアを可能な限り高速化する

今日ではディスクがかなり高速化されているので、通常のイメージ・コピー・ファイルのリストア自体はそれほど時間がかかるわけではありません。ESS(Enterprise Storage Server™)の見積もりとしてはページ当たり0.4ミリ秒であり、例えば2Gバイトのリストアは約3～4分で可能です。さらに高速化するにはフラッシュ・コピーなどのハードウェア機能の利用が考えられます。

(2)障害時にログの適応時間を可能な限り高速化する

- できるだけ頻繁にバックアップを取得し、障害時に適応しなければならないログそのものを減らします。
- ログの適応時間を高速化するツールを使用します。

技術的には業務を停止することなくデータベース・バックアップを取得することは可能です。ただし実際には、そのバックアップがデータの整合性を保つ静止点を持つ必要があるかどうかに左右されます。例えばあるバッチの更新が正しく行われなかった場合、その論理障害を起こしたデータベースの回復はコンカレントまででではなく、そのバッチがスタートする前のポイントまで戻さなくてはなりません。また、ある業務をリスタートする場合も、データベースを障害の直前まででではなく事前のある静止点に戻す、つまりバックアップ・ファイルをリストアしただけでログを適応しないという回復方法が妥当な場合もあります。このように一口にデータベース回復といっても、業務によってデータベースの回復要件は幾つかの条件が混在したものとなり、それぞれの要件により方法を検討する必要があります(図1参照)。

また、バックアップの取得方法はDBMSの機能、例えばDB2のイメージ・コピーが使用可能ですし、OS側の機能(DFDSSのコピーまたはダンプなど)、ハードウェア上の機能なども活用できます。頻繁にバックアップを取ることで回復時間を短くするならば、その機能への要件は次のようになります。

- 業務を止めずにバックアップが取得可能であること。
- できるだけ運用が簡単であること。
- 静止点を持ったバックアップが必要な場合は、その要件を満たす機能であること。

これらの要件を満たすかどうかを、現時点の各バックアップ取得方法とともにまとめたのが表2です。

DB2では従来からオンライン・コピー(COPY SHRLEVEL CHANGE)により、更新業務を停止することなくバックアップ取得が可能です。ただし、オンライン・イメージ・コピー・ファイルはアンコミット・データが含まれている可能性があるため静止点を持たないバックアップであり、TOCOPY指定のリカバーに使用できません。これに対して、CONCURRENT COPY SHRLEVEL REFERENCEによるバックアップは、実行時の最初に論理セッションが確立されるまでの数秒～十数秒間はリード・オンリーとなりますが、その後はアプリケーションからの更新

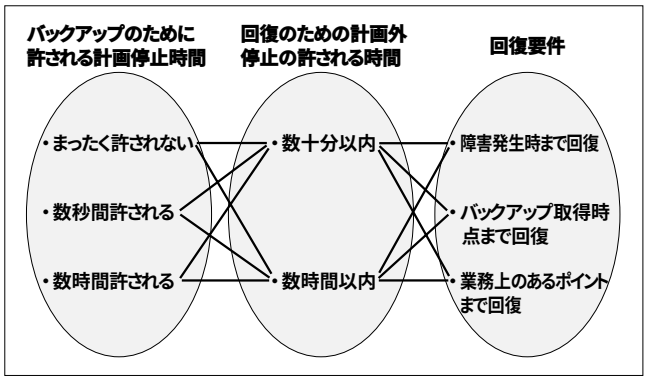


図1.データベースの回復要件

表2.バックアップの取得方法

| | 取得時に業務の更新が可能か? | 静止点回復が可能か? | バックアップ取得単位／管理対象 | 回復時運用容易性 |
|---|---|---|---|---|
| DB2コピー SHRLEVEL REFERENCE | 不可 | 可能 | ページ・セット単位 DB2により管理 | ◎ |
| DB2コピー SHRLEVEL CHANGE | 可能 | 不可 | ページ・セット単位 DB2により管理 | ◎ |
| DB2コピー CONCURRENT SHRLEVEL REFERENCE (注1、2) | 可能 (ただし数秒待機) | 可能 | ページ・セット単位 DB2により管理 | ◎ |
| フラッシュ・コピー DB2/390 V6以前 | 不可 (本文参照) | 可能 | ボリューム単位 非DB2管理 | △ |
| フラッシュ・コピー DB2/390 V7 | 可能 | 不可 | ボリューム単位 非DB2管理 | △ |

が可能です。その一方で、データ・セットに対してはQUIESCEが入るため、きちんとした静止点を持ったバックアップを取得できます。つまり、更新業務への影響が最小限であり、かつデータの整合性が保証されたバックアップを持つことができるという点で、この機能は非常に有効です。また、コンカレント・コピーの機能自体はDFSMS環境の設定が必要ですが、DB2の表スペースそのものはDFSMS環境配下にある必要はありません。

そのほか、ESSのフラッシュ・コピーを用いれば、DB2の管理外でDASDのハードウェア機能によりバックアップを取得することも可能です。この方法の特長は、バックアップ自体が数秒で終了することです。ただし、取得時にSTOP DBコマンドまたはQUIESCEユーティリティーにより、DB2バッファー内から更新データをいったんデータ・セットに書き戻させ、表スペースであるVSAMをクローズする必要があります。ハードウェア上の機能としてフラッシュ・コピーは、VSAMクローズをしなくても（すなわちDATASET IN USE状態でも）バックアップの取得が可能であり、特に静止点を持つバックアップを必要としない場合はQUIESCEさせないことも考えられます。ただし、この方法でバックアップを取得した際に、カレント・ポイントまでのデータを回復するには、バックアップ・ファイルをリストアした後に、DB2のログ・オンリー・リカバーを行う必要があります。このときログ・アプライのスタート・ポイントは、この表スペースのヘッダーに存在するHPGRBRBA内のRBA値となります。この値はV6まではVSAMクローズ時または論理クローズ時にしか更新されなかったため、上記のようにVSAMクローズしないでバックアップを取得すると、その時点でのHPGRBRBAは更新されません。その結果、ログ・アプライ時のスタート・ポイントはずっと前になり、回復に思いがけない時間がかかる可能性がありました。V7からはチェックポイント時に更新されるようになったため、フラッシュ・コピーによりデータベース回復を行うことが可能となりました。このようにハードウェア機能を活用する場合も、ソフトウェア側の機能により有効活用できないこともあり、注意が必要です。また、このようにDB2の管理外でバックアップを取る場合は、回復時にどのファイルが正しいバックアップであるのかをユーザー側で判断しなければならないため、それをいかに行うかが運用上の問題となります。

さらに、DB2システムとデータベースすべてのバックアップをフラッシュ・コピーによって一度に取る場合は、LOG SUSPENDコマンドが有効です。このコマンドによりDB2上のすべての更新業務を待機させ、その時点でDB2カタログ、アクティブ・ログ、BSDS、必要なユーザー・データベースの載っているボリュームのバックアップ取得をフラッシュ・コピーにより一挙に数秒で行います。終了後にLOG RESUMEコマンドを実行すれば、待たされていた更新業務は稼働可能となります。このときのバックアップは、リストア後にDB2の再始動を行うことで、たとえバッファー・プールからフラッシュされなかったデータが存在していても、再始動時にログにより回復させることができ、データの整合性は守られます。この方法は、災害対策システムへの移動や、小さな容量のデータ・セットが多量に存在するDB2（例えば、SAPなどのERP製品を使用しているDB2など）の静止点を持ったバックアップとして有効です。

**《ログの適応の高速化》**

データベース回復時に問題となるのは、ログ適応の時間がどのくらいかかるのかが事前に分からないことです。更新がランダムに行われたのか、あるいは同一データが繰り返し更新されたのか、更新状態によって時間が大幅に異なってくるからです。なぜなら、ディスクのI/O時間は例えばESSではシーケンシャルにスキャンする場合、4Kページ当たり0.4ミリ秒ですが、ランダム処理でキャッシュ・ヒットしない場合は、約10ミリ秒もかかると想定されます。ログ適応時ログ・リードはこのシーケンシャルの値で高速にリードできますが、適応側でランダムとなる場合はログ適応に長時間を要する可能性があります。当然ながら、ログをデータの並び順にソートして適応した方が時間を短縮できます。DB2/390 V6で拡張されたFLA（Fast Log Apply）機能はこのための拡張であり、回復時にログ・アプライのフェーズでバッファー内（最大10Mバイト）にログをいったんため込み、OBID、DBID、RBAなどでソートしてから適用します。これにより同時点で同一ページへの更新や、同一データへの更新が複数ある場合などにログ・アプライの時間短縮が期待できます。

ログ適応の高速化のためには、さらに次のような機能が考えられます。

- 最も新しく更新されたデータのログのみを事前に抽出し、データの並び順にソートしておく機能。
- データベース回復後に索引を再作成すれば、索引のログ自体がいらないため、索引のログを削除してログ量自体を減らす機能。
- 表単位にログを分割しソートしておくChange Log Accumulation機能。
- バックアップ・ファイルに事前にログを適応しておき、障害時にはそのファイルをリストアし、回復時に適応するログ量を大幅に減少させる機能。

ただし、上記の機能は残念ながらDB2ファミリーの標準機能には含まれておらず、DMツールまたは他社製品を使用する必要があります。また、各表単位で細かな運用を行うことから、

通常時の運用負荷が非常に大きくなることを考慮しなくてはなりません。

### 4.2. データベースの2重化

RAID5であるRVAやESSなどは、ハードウェア的にはその構造から1シリンダー単位やボリューム単位で障害が発生することを考慮する必要はほぼなくなっています。ただし、問題はRANKおよび筐体全体が停止する場合です。もちろん筐体障害が発生する確率は非常に小さいのですが、ESSのように1筐体に非常に大きな容量を保持している場合、いったん障害が発生するとその影響は非常に大きく、データベース回復にはかなりの時間がかかるでしょう。そのため、重要性が非常に高いデータベースについては筐体をまたがって2重化するべきです。

**《アプリケーション上での2重化》**

残念ながら、現在のDB2ではIMS/DBのようにDBMSとしてデータの2重書きを行う機能をサポートしていません。そのため、アプリケーション上で仕組みをつくることを検討しなくてはなりません。その場合、当然ながら1回の更新業務により二つの表の同期を取って更新しなくてはなりません。実行時間への影響は、SQL実行のためのCPU負荷とDB2上のログの書き込みコストなどが主ですが、今日のように高速なCPU速度においては実際の実行時間が延びるようなことはあまり考えられません。ただし、プライマリーのDASD障害時にセカンダリーのみの更新にどのように切り替えるのか、バッチ処理やLOAD、REORGといったユーティリティーの運用をどのように行うのかなど、多くの仕組みもつくらなくてはならず、現実的とはいえないでしょう。

**《ハードウェア機能による2重化》**

ハードウェアを使用してDASDを2重化した場合、理想的な2重化の要件と、現在のESSが保有しているコピー機能であるPPRC、XRCの二つの機能の比較を表3に示します。

理想的な2重化機能は、2重化のパフォーマンス悪化がほとんどなく、さらにプライマリーのDASD障害発生時には自動的にセカンダリーに切り替えが発生し、エラーにより業務が中断することもありません。データベースの回復も必要がありません。もちろん筐体障害に備え、2重化のペアは筐体が異なる状態で行えます。

この要件に比較すると、まずXRCは、プライマリー側のDASDへの書き込みデータをキャッシュ経由で非同期にセカンダリーにコピーするため、プライマリー側のパフォーマンスにはほとんど影響しません。ただし非同期コピーであるが故に障害時に数秒間のデータ・ロスト発生の可能性があります。プライマリー側DASDが障害を起こした場合は、セカンダリー・データベース側でそのデータをカレントまで戻すため、RECOVERユーティリティーによる回復が必要です。つまり、せっかく2重化しても回復に長時間がかかるのでは、データベースの2重化という観点からはあまり有効ではありません。さらにこの機能では、セカンダリー側に別のOS/390を立ち上げ、SDMという管理ソフトウェアを動かすことが必要です。そのための資源が必要であり、データベースの2重化よりも災害対策システムの機能として有効であると考えられます。

PPRCは、同じくキャッシュ経由でデータをコピーします。こ

表3.PPRC、XRCの比較

（P：プライマリー・ボリューム　S：セカンダリー・ボリューム　注：◎○△×は評価）

| | 理想的なデータベース2重化 | PPRC | XRC |
|---|---|---|---|
| 更新反映の同期／非同期 | 同期 | 同期 | 非同期（最少で1秒間隔） |
| P障害発生時 | アプリケーション側にエラーが戻らない | デバイス・エラー発生　×　DB2は－904を返す | デバイス・エラー発生　×　DB2は－904を返す |
| P障害時手順 | 自動的にスイッチ | 手作業でオフライン／コマンド実行　× | 手作業でXRECOVER　コマンド実行　× |
| 通常時パフォーマンス | 悪化なし | I/O実行時間に影響　△<br>（ただしダイレクト接続時。通信回線経由ではもっと悪化する） | 悪化の度合いは低くなる。　○<br>（ただし書き込み量に見合う通信速度とキャッシュ容量が十分な場合） |
| ペア構成 | 別筐体 | 別筐体　◎ | 別筐体　◎ |
| 構成・環境 | できれば制約なし | チャネルが必要　○ | SDMが稼働する別OSが通常の場合必要<br>CDSやジャーナルが必要　チャネルが必要　△ |
| S障害時 | 影響なし | 影響なし　◎　（設定により可能） | 影響なし　◎（設定により可能） |
| DB2のリカバリー | 必要ない | START DBコマンドによりLPL回復させる必要あり　△ | RECOVERユーティリティーによる回復作業が必要　× |

ちらはチャネル経由でダイレクトにESS同士をつなぎ同期を取ります。そのためI/O時間に影響を受けます。また、プライマリーDASD障害時にはI/Oエラーが検知されるため、例えばDB2のバッファー・プールから書き出されたページなどはI/Oエラーとなり、いったん書き込み障害状態（LPLステータス）になります。しかしこのLPLは、STARTコマンドによってログのみから短時間で修復が可能であり、PPRCの場合はセカンダリーに切り替わった後、このLPLのみを修復すれば業務を続行できます。

**表4．DB2によるソフトウェア上の2重化機能の要件**　P：プライマリー側データベース　S：セタンダリー側データベース

| 要件項目 | サブ項目 | 要件 |
|---|---|---|
| パフォーマンス要件 | リードI/O | 同期・非同期I/O： 2重化に影響されない |
| | ライトI/O | •同期I/O： ELAPSEDタイムへの影響は2倍以内<br>•非同期I/O： バッファー・プールからの書き出しはアプリケーションには影響しない<br>•バッチに関しては十数パーセント以内の悪化に収める |
| | ユーティリティー | 非同期I/O： 十数パーセント以内の悪化に収める |
| | ログ | ログの書き出し量は2重化によって増えない |
| DB2内での認識方法 | テーブル・スペース | •異なる任意のテーブル・スペース名（VSAMクラスター名）をペアであると定義<br>•もちろん同じ属性であることが前提<br>→検知機能が必要<br>•2重化するかどうかは選択可能<br>VSAMの第1修飾子が異なったものをペアにできる（異なったUCATにしたい）<br>VSAM名も変えられる。ただし何らかのネーミング・ルールは必要<br>ユーザー・マネジメント・データ・セットはどうするか? |
| | インデックス | •異なるインデックス名（VSAMクラスター名）をペアであると定義<br>•もちろん同じ属性であることが前提<br>•同じ表の索引でも2重化するかは選択可能 |
| | 区分単位 | •区分表は区分単位で2重化するかどうかの選択を行えるようにする<br>•区分索引も同じ<br>•非区分索引は2重化するかを選択可能 |
| | その他 | •表名、その表に対するプラン、パッケージは同一に認識し管理<br>•PからSにスイッチした際にバインド、再バインドは行わなくてよいこと<br>•アクセス・パスも変更されないこと |
| 障害時検知 | P側障害 | ライト側<br>以下の機能を選択可能とする<br>（1）即Pを切り離しSにスイッチ<br>アプリケーションにはSQLエラー・コードは戻らない<br>DB2は検知し、DSNメッセージをSYSLOG上に出力<br>（2）もし可能なら、LPLがある程度発生したら自動的にスイッチ<br>THRESHOLD値を指定可能<br>（3）コマンドで切り替え<br>リード側<br>•Pを読むのをやめ、Sから再リードしSINPLEXに切り替える<br>•そのときアプリケーションにはSQLエラー・コードは戻らない<br>•DB2は検知し、DSNメッセージをSYSLOG上に出力 |
| | S側障害 | •P側アプリケーションにエラーは一切起こらない<br>•DB2はSYSLOG上にメッセージ出力し、SINPLEXに切り替える |
| 2重化ステップ | | •ストレージ・グループはPとSで分ける（別筐体に配置可能にするため）<br>•業務稼働中にDUPLEXに可能とする<br>（1）PからSにオンライン・コピー（SHRLEVEL CHANGEのコピー）<br>このときテーブル・スペースと2重化対象索引のコピーの同期を取る必要あり<br>（2）コピー中のPに対する更新は、コピー終了後ログよりSにアプライ、オンラインREORGの要領などで。 |
| その他 | | •COMPARE CHECKユーティリティー<br>PとSのデータが正しいかどうかをチェックするユーティリティーが必要。<br>•回復<br>生きている方をリカバー可能<br>イメージ・コピーは通常Pから行い、リカバー時にはP、S両方で使用可能。ログもPでもSでもアプライ可能に。<br>•コマンド<br>DISPLAY DBでDUPLEXかどうかを確認できる |

上記二つの機能のうち、筐体障害に対応するデータベース2重化を行うハードウェア機能としては、現時点ではPPRCが有効と考えられます。ESSの異なる筐体を通信回線経由ではなくダイレクトにチャネル接続した場合、同期I/Oの応答時間は2倍になりますが、DB2の更新パフォーマンスは、アプリケーション側においては「バッファー・プール上での書き込み＋ログI/O時間による書き込み」であり、実際のDASDへの書き込みI/Oはバッファー・プールがあふれたときなどを除いてもともと非同期なので、トランザクション・タイプのアプリケーションへのパフォーマンスの影響は少ないと考えられます。また、テスト結果では、バッチやユーティリティー・ジョブに関する影響も、チャネル直結ならば10%ほどのパフォーマンス低下であり、現時点ではPPRCは2重化の観点では唯一有効な手段といえるでしょう。さらに今後、PPRCに関してはI/Oエラーを返すことなくセカンダリーにスイッチする機能拡張などが予定されています。実際には他社製品などの機能比較も行う必要がありますが、上記要件を当てはめて検討すべきと考えます。

### 《DB2による2重化》

先に述べたように、現時点ではDB2ファミリーとして、データベースの2重化機能をサポートしていません。PPRCのようにハードウェア機能で2重化する方が運用的にも簡単ですが、筐体障害などはもともとハードウェア上やマイクロコードなどの問題により発生するため、その際に2重化機能も巻き込まれる可能性を考慮すると、やはりDBMS側のソフトウェア上の2重化機能が今後は必要と思われます。その機能を想定して要件をまとめたのが表4です。

以上のように、DBMS側で2重化を行うと、ユーザー側運用面に関してはさまざまな考慮

点が生じてきます。しかし機能的な柔軟性を考慮すると、ハードウェア上で行うよりも優れていますし、応用もできるでしょう。2重化の機能に関しては、既に多くのお客様から機能拡張の要求が出されており、IBMとしては早急に対応すべきと考えています。

## 5. データベースの計画停止に関する考察

### 5.1. ユーティリティー

データベースの計画停止の要因として、データベースは順次メンテナンスを行わなくてはならないという点があります。いったん作成した後、ある程度の期間ごとにデータ変更・パフォーマンス・容量などの観点から表5に示したようなユーティリティーを実行する必要があります。DB2/390上でそれらのユーティリティーが連続稼働の観点でどのような影響を持つかについても表にまとめました。

以上のように、ユーティリティーに関してはこれまでの機能拡張により計画停止はほぼ回避可能になっています。

特に、今までかなり長時間にわたってデータベースを止める必要のあった再編成処理では、DB2/390 V5よりオンラインREORGが可能となっています。オンラインREORGは、極短時間アクセスがR/Oになったり待機になるフェーズはありますが、業務側はその間待機するだけであり、オンライン中にも再編成ができる点は24時間連続稼働に有効です。

ただし、LOADなどはその機能上、表に対する業務を停止することになります。ロードしながらCOPY、RUNSTATSを行ういわゆるINLINE COPY、INLINE RUNSTATSや、V7のONLINE LOAD RESUME機能（V7）、さらに表を区分化して区分単位でLOADを実行するなど、LOADユーティリティーの実行時間を短縮することが可能となっています。しかし実際には、LOADはデータベースの計画停止を必ず行わなくてはいけないので、LOADユーティリティーを使わないか、あるいはLOADユーティリティーによるデータベース停止をほかに影響させない方法を検討する必要があります。この点に関してはアプリケーション側の停止回避の方法で検討します。

表5.連続稼働から見たユーティリティーの評価

| ユーティリティー | 評価 | 備考 |
|---|---|---|
| LOAD REPLACE | × | その表に関しては業務停止 |
| LOAD RESUME YES | × | その表に関しては業務停止 |
| ONLINE LOAD RESUME<br>INLINE COPY、INLINE RUNSTATSが可能 | △ | V7の新機能（RESUMEのみ） |
| REORG SHRLEVEL REFERENCE | × | アクセス不可 |
| REORG SHRLEVEL CHANGE<br>INLINE COPY、INLINE RUNSTATSが可能 | ○ | フェーズごとに待機する場合があるが基本的にリード／ライトは可能 |
| COPY SHRLEVEL CHANGE | ○ | リード／ライト可能、パラレル・コピー（V6） |
| CONCURRENT COPY SHRLEVEL REFERENCE | ○ | 最初の十数秒はR/O以降はリード／ライト可能 |
| COPY SHRLEVEL REFERENCE | △ | リード・オンリーのみ |
| RUNSTATS SHRLEVEL REFERENCE | △ | リード・オンリーのみ<br>LOAD、REORG時に同時にコピー取得 |
| RUNSTATS SHRLEVEL CHANGE | ○ | リード／ライト可能 |
| MODIFY | ○ | RECOVERなどのユーティリティーとの同時実行は不可 |

### 5.2. 変更作業に関する考察

変更作業としては以下の項目が存在します。これらの変更も計画停止を行わないで可能かどうかを検討します。

- データベース属性の変更
- プログラムの変更の反映
- システム・パラメーターの変更
- システム保守（APARなどの適用）
- バージョン・アップ

#### 5.2.1. データベース属性の変更

DB2/390において既存の表の属性などを変更する場合、表6に示すように停止することなく変更可能な場合と、DROP／CREATEにより作り直しが依然として必要な場合があります。

DROP／CREATEが必要な変更は、表の構造そのものの変更であり、DROP／CREATE時における表に関する業務停止、およびその表をアクセスするパッケージやプランの再バインドが必要となるなど大きな影響があります。

しかも変更対象となる表への影響のみで済まされない場合もあります。例えば、物理設計時点でデータベースの定義を考慮する必要があります。DB2/390においてデータベースという属性は論理的なグループを示し、物理的な意味を持たない集合ですが、同一のデータベース内に複数の表スペースを登録すると、それらの表スペースは同一のDBDを持ちます。このとき表スペースに対してDDLを実行すると、DBDにX-DBDロックが掛かり、DML実行時にはS-DBDロックが取得されます。よって、例えば既存のデータベースに新しい表スペースなどを追加する場合、同じデータベース内のほかの表スペースに対して業務が

表6.データベース属性の変更

| データベースの停止することなく変更できるもの | データベースのDROP／CREATEが必要なもの |
|---|---|
| • ALTER INDEX PARTI/ON RANGE（V6）<br>• 区分のキー範囲変更。ただし対象の区分はREORGが必要。<br>• ALTER PRIQTY、SECQTY、FREEPAGE、PCTFREE、CLOSE<br>• ALTER VARCHAR COLUMN（V6）<br>• ADD COLUMN | • 表の中央に列を追加<br>• 表から列を削除する場合<br>• 索引を構成する列を変更する場合<br>• 列定義を変更する場合<br>• 非区分表を区分表に変更する場合<br>• 区分の総数を変更する場合<br>• テーブル・スペースをLARGE指定に変更 |

行われているとロック待ちが生じることになります。業務停止を行うことなく変更・追加を行うには、一つのデータベースに関連外の資源をあまりたくさん登録しないことをお勧めします。

このようなシステム上の問題に関しては、早い段階からのお客様に対する十分なガイドが必要となります。

- DROP／CREATEを伴うような変更はデータベースの論理設計レベルからの変更となるため、開発段階からこうした変更が後から生じないように注意します。
- 物理設計段階から可用性を高める設計を盛り込むように配慮します。

### 5.2.2. プログラム変更のための考察

DB2上では、各プログラム・モジュールは「パッケージ」という単位で認識されています。プログラム変更が発生したときにバインド・パッケージを行う必要があり、業務を停止することになる場合があります。これはプランやパッケージの実行時にはSKELETON CURSOR/PACKAGE TABLEにSロックを取得し、バインド時にはSKELETON CURSOR/PACKAGE TABLEにXロックを取得するため、アプリケーション実行時に使用中のパッケージをバインドすることはできないからです。

よって、常駐タスクのように業務停止時間をつくるのが難しい業務については特にバインドのタイミングを考慮する必要があります。

また、e-ビジネスのような3層型システム構造の場合、アプリケーション・サーバーが複数あるため、一度にプログラム変更を行えない場合も想定されます。その場合、同じパッケージ名で異なるバージョンのプログラムが稼働する必要が生じます。このようなときは、DB2/390ではパッケージのバージョンID機能を使用すると、業務稼働中にパッケージの追加などが可能となります。例えば、現在実行中のパッケージはPCK1（バージョンA）でバインドされているとします。プログラム変更が発生し、変更後のパッケージをPCK1（バージョンB）でバインドします。これによりパッケージ名は同一ですが、バージョンが異なったアプリケーションを実行可能です。その後、例えばCICSプログラムならば、NEWCOPYコマンドによりルーチン・モジュールを変更前から変更後にリプレースさせます。そのときDB2はこのルーチン・モジュール内のバージョンをチェックし、同時にPCK1（バージョンB）を使用します。これにより、業務停止することなくプログラムの変更を行うことが可能です。このバージョン機能はDB2 UDB FOR AIX側もV8よりサポートされる予定です。

もしプログラム内の変更がSQL部分でない場合には、モジュール内のTIMESTAMPを変更しないでコンパイル可能なバインド・マネジャーなどのツールもあり、それらの使用も検討すべきです。

### 5.2.3. その他

《システム保守》

障害発生を事前に予防するには、APARの適用を随時行う必要があります。従来、APAR適用時にはほとんどの場合で業務を停止する必要がありましたが、シスプレックス上の機能の拡張により、390上ではOS/390、DB2を問わずに異なるPTFレベルで各メンバーが稼働可能であり、システム保守による全面停止は回避可能になっています。

《システム・パラメーター（ZPARM）の変更》

ZPARMのパラメーター変更に関しては、まずV6でSET LOGコマンドにより動的にLOGLOAD値（CHECKPOIN頻度）が変更可能となりました。さらにV7によって、EDMプールやDSMAXなどのほとんどのZPARMが動的に変更可能となっています。

《マイグレーション・インパクト》

DB2/390自体のバージョン・アップ時にはCATMAINTによるDB2カタログ／ディレクトリーの変更作業を行うために業務停止が必要です。今後の機能拡張が待たれるところです。

## 5.3. アプリケーション障害に対する考慮点

以上、DBMS側の機能に関して検討項目を挙げてきましたが、アプリケーション側でも高可用性のための検討項目が多くあります。連続稼働を可能にするには業務設計の初期の段階から、連続可用性を妨げる要因を意識してください。

### 5.3.1. ホット・スポットの回避

アプリケーション障害またはトランザクション障害が全面停止にまで波及するのを回避するには、多くの業務が一つの表に集中してアクセスするようなホット・スポットをできるだけ作らないように設計することです。なぜなら、高トランザクション環境ではホット・スポットでのデッドロックやタイム・アウトが発生することで資源不足が生じ、全面業務停止の引き金になる可能性があるからです。データベース上でのホット・スポットとしては、コード表・プロダクト表など数ページしかない小さな表に集中的にアクセスするホット・ページのケースと、採番表・カウンター表など1表の中に1行しかデータがなく、それを更新し合うようなホット・ロウのケースがあります。表の分割または区分化などを行うことで、データベース物理設計時に対応可能か、さらにグループ採番の採用などにより業務的に変更が可能かを検討します。

### 5.3.2. オンライン業務とバッチ業務の共存に関する考察

従来、夜間にはオンライン業務を停止してバッチ業務を行う

という運用が一般的でしたが、今日ではオンライン業務をできるだけ24時間停止させないで運用したいという要件が増えています。この要件に対応するには次のような方法が考えられます。

(1)同一表上でオンラインとバッチを共存させる場合

更新バッチ業務を、オンライン側にできるだけ影響を与えない形で運用します。そのためには以下の点を考慮します。

- UNLOAD／LOAD処理をSQL更新処理に変更する

いったんデータをアンロードし、SAMファイルに落として業務処理してロードを行っていたようなバッチは、UPDATE／INSERT処理に変更することを検討します。

- ロックとコミット頻度を考慮する
  - ― バッチ側で頻繁にコミットを行い、オンライン側でロック・タイム・アウトを発生させないようにします。
  - ― バッチ実行時にロック・エスカレーション、ロック・プロパゲーションを起こさないように、アクセス・パスやアイソレーション・レベルに注意します。ただし、コミットを頻繁に行うと、当然ながらバッチ・ジョブ全体の実行時間・CPU時間が増大します。よって、従来夜間に大量に行っていた更新バッチを小さく分割し、いわゆるCPUバッチという形態で1日の間に分散し、夜間のバッチ業務時間そのものを短縮している事例もあります。

(2)バッチ処理中オンライン業務のデータ追加を別表に行う方法

夜間のオンライン業務が受け付け処理のように、データベース的にはインサート処理である場合は、図2に示すようにオンライン業務の対象表を別表に切り替える方法も考えられます。

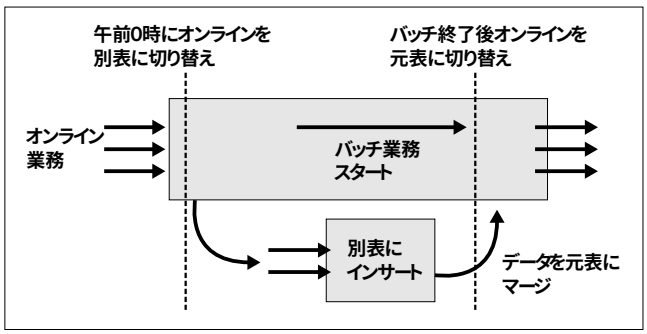


図2.バッチ処理中オンライン業務のデータ追加を別表に行う方法

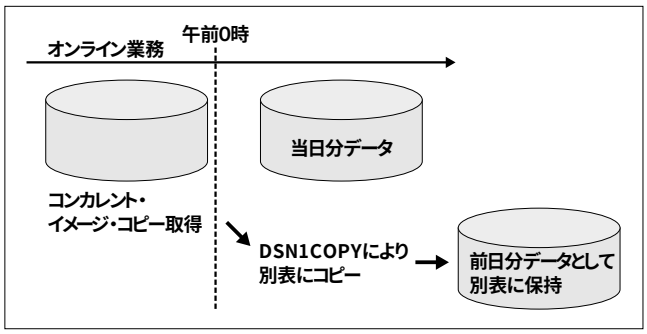


図3.ある表のデータを前日分として別表にコピーする方法

(3)ある表のデータを前日分として別表にコピーする方法

ある表のデータを前日分としてコピーしておき、業務処理を行うか、または情報系検索などに活用する場合は、イメージ・コピーをコンカレント・コピーで取得し、それを別表にコピーすることも可能です。前述したように、コンカレント・コピーは業務側に更新・検索を許し、静止点を持ってバックアップが取れるという特性を生かすことで図3に示すような処理が可能となります。

以上のように別表にレプリカを行う方法であれば、バッチとオンラインの双方が共存してもロックやコミット頻度を考慮する必要はありません。ただし、どのようにレプリカを行い、どのようにデータ・マージを行うかは、その業務形態により不可能な場合もあり、その観点から十分に検討すべきです。

## 6. おわりに

以上のように、24時間連続稼働を実現するには、システム上の機能面、アプリケーション面、データベース設計などの点においてさまざまな考慮が必要であり、多くの困難が存在します。確かにDB2/390だけではなく、ハードウェア機能やOS/390上の機能も「業務を停止させない」または「停止する時間をいかに短くするか」の観点でさまざまな拡張が行われていますが、まだ十分とはいえません。

さらに、常時資源をモニタリングするための運用コストやデータベースの2重化、ハードウェア上の機能の活用に際しては多くの投資が必要であることも明らかです。しかし、これらは今後必要な要件であり、また完全ではなくても そのようなシステム上の機能を上手に組み合わせてアイデアを実現することで、24時間連続稼働を実現したシステム構築することは可能と考えられます。

要求のますます高まる高可用性システム構築に対して、本論文が少しでも活用されれば幸いです。

(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] Frank McGrath「Hey Dad, Why Can't We Go 24 by 7?」IDUG 2000資料