DB2大規模基幹系システム構築アプローチ

～DB2はTPF・IMSFPを超えられるか～

1,000件/秒以上の処理能力を持つ大規模基幹系システムの構築において、従来の基幹系に特化したTPFやIMSFPでのアプローチでは、開発・維持管理のコストはますます増大する傾向にあります。このためベストセラーDBMSであるDB2®による大規模基幹系基盤構築の実現を望むお客様が増加してきました。DB2で、従来の座席予約システムや都市銀行勘定系などに求められる高処理効率・高信頼性を実現できれば、TPFはもちろんIMSFPよりも開発生産性が高く、DB2の持つ自動ナビゲーション方式によりデータ変動にも強く、さらにオープン環境との親和性の良さによりクライアントからの要求にも短期間で対応可能なシステム基盤となり得ます。

本論文では、国内を含むアジア太平洋地域の国々において徐々に実現されつつあるDB2による超大規模・高信頼性の基幹システム構築（一部は既にサービス・イン済み）について、そのサポートを通じて得られた課題点克服のためのアプローチ方法を議論していきます。例としてzSeries™＋DB2 for z/OS™環境を取り上げ、都市銀行勘定系を含めた大規模基幹系構築を中心に効率の良い大量トランザクション処理および高信頼性システム実現に必要とされる実践的な設計方法について検討を加えています。

日本アイ・ビー・エム システムズ・エンジニアリング株式会社
ITソリューション・センター
ICP-シニア・コンサルティングITスペシャリスト
Senior Consulting IT Specialist
IT Solution Center
IBM Japan Systems Engineering Co.,Ltd.

西川 善夫 Yoshio Nishikawa

[プロフィール]
1977年、日本アイ・ビー・エムに入社。営業所SEとして家電メーカーを担当。1987年、テクニカル・サポートへ。
現在、並列シスプレックスおよびDB2/zOSシステムのAP（アジア太平洋地域）のカントリー・サポートに従事。

Approaches to the Construction of a DB2 Large-Scale Mainstay System --Will DB2 be able to overtake TPF and IMSFP?--

In the construction of large-scale mainstay systems with a processing capacity of more than 1,000 tasks a second, there is a tendency for development and maintenance control costs to rise to ever higher levels on the basis of an approach with TPF and IMSFP aimed specifically at existing mainstay systems. For this reason there has been an increase in the number of customers hoping to see the construction of a large-scale mainstay system infrastructure employing DB2®, the best-selling DBMS. If it becomes possible to achieve a high level of processing efficiency and reliability as is required of existing seat reservation systems and city bank accounting systems with DB2, development productivity will be superior not only to TPF but also to IMSFP. This may well become a system infrastructure fully capable of coping with changes in data due to DB2's automatic navigation system and one that will be able to respond in a short time to demands from clients thanks to the good affinity with open environments.

In this paper we take a look at the construction of the ultra-large, highly reliable mainstay systems based on DB2 that are gradually being realized in Japan and throughout the Asian and Pacific area (some of which have already entered service), and in particular at the approaches being adopted in order to get the better of the various problems that have been acquired through support. By way of example, we look at the zSeries™ + DB2 for z/OS™ environment and examine the practical design methods that are regarded as necessary for realizing highly efficient, large-scale transaction processing and highly reliable systems centering on the construction of large-scale mainstay systems including the accounting systems of city banks.

1. はじめに

座席予約システムを実現するTPF（Transaction Processing Facility：トランザクション処理機構）や、都市銀行勘定系の中核となるIMSFP（Information Management System Fast Path：IMS高速パス）レベルの高速処理・高信頼をDB2®で実現するには、多くの課題の克服が必要とされてきました。

しかし現実には、既に情報系においてDB2環境でも2,000件/秒以上の高速更新メッセージ処理システムが稼働中であり、都市銀行勘定系でも韓国では既に500件/秒規模の大規模のシステムが並列シスプレックス＋DB2 for z/OS™環境で本番稼働しています。また1,200件/秒の勘定系システムのサービス・インが予定されています。もちろんこの新しいシステム基盤の実現には、考慮すべき重要な設計ポイントが数多く存在します。

本論文では、筆者が実際に設計に携わった多くのDB2大規模システム構築を通して得られた設計アプローチ方法を、現実の課題と解決方法とを対比させながら検討を加えていきます。

2. DB2大規模基幹系構築の課題

2.1. 正確なキャパシティー予測の必要性

1,000件/秒を超える高速更新トランザクション環境ではプロセッサー、主記憶装置、DASD（Direct Access Strage Device）、ネットワーク機器などのハードウェア資源はもちろん、ECSA/CSA（Extended Common Storage Area / Common Storage Area：拡張記憶サービス域／記憶サービス域）や16Mバイト以下の仮想記憶空間、DB2更新ログなど、ボトルネック要素になり得る資源に対し大きな負荷がかかります。当然、単一SCP（System Control Program：システム制御プログラム）やDBMS（Database Management System：データベース管理システム）の処理限界を超える場合も発生します。大量トランザクション処理によるCPU（Central Processing Unit：中央演算処理装置）コストや関連I/O（Input/Output：入出力）コストだけではなく、センター・カット（公共料金やクレジットの預金者口座からの一括引き落としなどの大規模バッチ処理）のコストや、同時オープン・データベース数、最大ロック保持数（仮想記憶域必要量に影響）などにも注意が必要です。

TPFやIMSFPなどのある程度制約された開発環境と異なり、DB2環境では設計の自由度が高いため、早い段階から本番を想定した代表的トランザクション・パターン分析やバッチ運用形態を意識して負荷を予測しておかないと、サービス・イン後に資源不足やボトルネックが顕在化する場合が多くあります。

負荷予測についても、設計段階が進むにつれて明確化する変動要素をモニターし、短時間で新たなシステム資源必要量を予測修正しながら進むことのできる体制を事前につくり込む必要があります。ベースとなる必要資源が非常に大きいため、修正に伴う構成変更が大規模になる可能性があり、継続したモニタリングが重要となります。また仮に初期段階での予測に対し200～300%程度の想定外の追加負荷が生じても、システム基盤や基本設計自身を見直すことなく、CPU、I/Oなどのハードウェア資源の追加だけで対応可能なシステム基盤環境を提供することも重要です。

実際に、統合テストの段階で発表された銀行間の合併により、最終的な更新メッセージ量が約2倍に増えた場合でさえ、事前のシミュレーションで求められた基本処理コスト（以下、原単位）に基づくハードウェアの増強だけで、ボトルネックを発生させることなく想定外の増加負荷を吸収した例も存在します。それには原単位とボトルネック要素を事前に正確に予測することが非常に重要であり、対象はDB2環境、ネットワーク送受信、MQ（Message Queuing）、CICS®、データベース共用コストなど多岐にわたります。この部分は自由度の高いDB2環境での開発で特におろそかになりがちであり、注意が必要です。

サービス・イン後に発生する最大の問題要素は、不十分なキャパシティー予測に起因する資源不足であることを十分に認識する必要があります。

2.2. 自動ナビゲーションの光と影

従来のTPFやIMSFP環境では高速トランザクション・システム実現のためにデータベースのデータ構造を事前に十分に把握し、最も効率の良いデータ処理順序を設計者自身で決定しました。すなわち処理フローとデータを一体化することで高速処理を実現してきました。それに対しDB2は表の結合順序や索引選択などはDB2で自動的に行い、処理フローとデータの独立性を可能な限り高めることで生産性向上と維持管理コスト削減を実現しています。

業務環境の変化によるデータ特性の大きな変化にも処理効率を落とさずに耐えられるDBMS環境は、アクセス・フローを事前に指定するTPFやIMS™などの固定ナビゲーション方式ではなく、データ特性の変化に柔軟に対応できる自動ナビゲーション方式を採用したDB2により実現されました。当然、自動ナビゲーション方式はいかなるデータ環境でも最も効率の良いアクセス経路を選択することが求められます。データベース規模が小さい場合は、仮に不適切なアクセス経路が選ばれてもそれほど大きな問題になりませんでしたが、大規模データベース環境ではアクセス経路により処理コストが大きく変

背景：お客様データ規模の推移：　～1987年　700万件/表
～1990年　5,000万件/表
～2002年　10億件以上/表
→データベース大規模化に比例して、アクセス選択経路間でパフォーマンスに大きな差

例：

	表1(T1)貸し出し関連	表2(T2)預金関連
	行数：1億2,000万	行数：7億8,000万
	行長：72	行長：30
	ページ数：250万	ページ数：750万

ネスト・ループ結合操作：SELECT * WHERE T1. C1 = T2. C1

T1→T2順に結合した場合：
- T1スキャン　入出力回数=250万
- T2ユニーク索引アクセス　入出力回数=1億2,000万×2
  合計入出力回数=2億4,245万

T2→T1順に結合した場合：
- T2スキャン　入出力回数=750万
- T1ユニーク索引アクセス　入出力回数=1億2,000万×2
- T1索引上データ不在の確認　入出力回数=(7億8,000万－1億2,000万)
  合計入出力回数=9億750万

★結合順序による入出力回数の差＝約6億6,500万
通常は小さい表から大きい表の順序で結合するが、WHERE条件によってダイナミックに変動。→DB2は統計情報に基づき自動的に最適アクセス経路を判断。

図1. 正確なアクセス経路選択の重要性

参考：DB2自動ナビゲーションによるアクセス経路選択
例：3表の結合SELECT：表A・表B・表Cにはおのおの2個の索引が定義

表A ←索引IA1 ←索引IA2
表B ←索引IB1 ←索引IB2
表C ←索引IC1 ←索引IC2

■表Aへのアクセス方法：
1. 表スペース・スキャン
2. 索引IA1の使用
3. 索引IA2の使用
4. 索引IA1・IA2同時使用計4通り。表B、表Cに対してもおのおの4通り

■表A・表B・表Cの結合順序は6通り
A-B-C　A-C-B　B-A-C　B-C-A　C-A-B　C-B-A

■結合方式はおのおの3通り(NLJ／MJ／HJ)
→考えられるアクセス方式は...
4×4×4×6×3×3＝3,456通り
→この中から最も効率良いアクセス経路を選択

応用：索引が各表に3個定義されていると、単一表のアクセス方法は上記の4通りから表スペース・スキャン／索引IA1使用／索引IA2使用／索引IA3使用／索引IA1・IA2同時使用／索引IA1・IA3同時使用／索引IA2・IA3同時使用／索引IA1・IA2・IA3同時使用の計8通り
→　8×8×8×6×3×3×＝27,648通り!!から選択。
→　最適経路の発見は静的SQLの場合、実行時でなくBIND時に完了させ選択コスト最小化。

図2. アクセス経路選択肢パターンの重要性

化するため、最新のデータ特性を反映したカタログ統計値に基づくアクセス経路選択の精度が重要になります(図1)。

IMSなどのようにアクセス経路が最初から固定されていると、データ量や条件指定の変化に応じて経路変更を行うアプリケーション・プログラムの修正が必要になり、結果として維持管理コストが増加し、解決までの時間も長期化します。DB2の場合、本来なら図中の表1→表2というアクセス順序は最近の統計情報により自動選択されるのであり、事前にアクセス経路が固定されているわけではありません。すなわち処理フローとデータベース内のデータ特性が互いに独立しています。そのため処理効率の向上のためにはデータ特性が変化した場合でも、処理フローを修正することなく最新のデータ特性情報に基づいて、多数の選択肢から最適なアクセス経路を選択することが重要になります(図2)。正確な統計情報をいかに反映し、多様なSQL(Structured Query Language：構造化照会言語)パターンやホスト変数などの変動要素に十分に耐えられる環境を整えることが重要です。それにより開発生産性の向上に貢献するだけでなく、処理効率を限りなくTPFやIMSなどの従来の基幹系プラットフォームのレベルに近づけることが可能になります。

実際の基幹系業務ではキー指定での単一表アクセスが多く、上記の例のような結合操作はバッチ処理や統計作業などに限られますが、単一表アクセスでも正しい索引選択の必要性は高信頼性システムを構築する限り、依然として存在します。逐次変化するデータ特性を示すDB2カタログ内の統計情報と、実行SQLステートメントで指定したWHERE文節のホスト変数を含む条件を総合的に判断した最適な経路を選択することが重要です。これが不正確だと、大量トランザクション処理や多重実行される大規模なセンター・カット処理を、限られた資源環境の下で効率良く実行することが難しくなります。

アクセス管理コストを最小化するための処理フローのデータ特性からの独立と、TPFやIBMFPなどの従来システムに匹敵する処理効率を両立させることが常に求められています。

2.3. データ競合事前予測の困難さ

一般に基幹系システムは、情報系システムに比べて業務要件や処理データ内容、トランザクション量などが明確であり、業務処理での最大の懸案事項であるデッドロックやタイムアウトの原因となるデータ競合の予測は比較的容易といわれてきました。しかし過去の経験では、DB2基幹系システムの総トランザクション量が200件/秒を超えるころから次第にデータ競合に敏感な状態になり、1,000件/秒を超えるシステム設計では、データに対するアクセス集中を最初からかなり意図的に分散させないとデータ競合の抑止は容易ではありません。

さらに重要なのは、大規模基幹系の設計では最低でも10年程度のシステム・ライフが想定され、企業合併やインターネットの普及などにより、データ競合の事前の想定が大幅に狂うことが十分に考えられます(図3)。つまり、設計時に予測したアクセス分布がデータ分散度合いに基づいて競合を最小限に

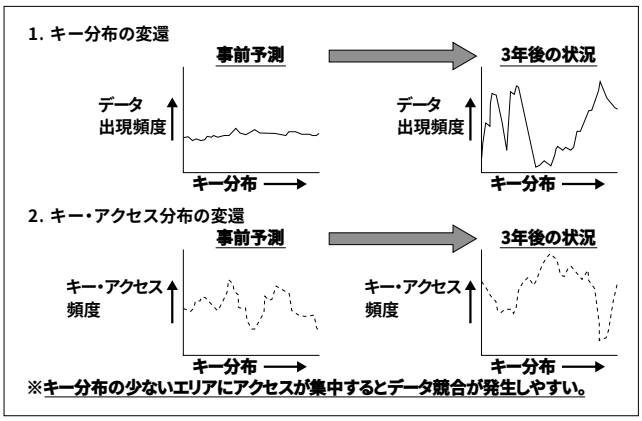


図3. 変化するデータ環境

抑制する工夫を施したとしても、データ特性の変化に対応しきれなくなる可能性は十分にあり得ます。またIMSFPではロック対象はCI（Control Interval）レコードですが、特定キーのルート部分を押さえることによりデッドロックはある程度まで防止できます。一方、DB2には基本的にルート・セグメントという考え方がなく、その点でもデータ競合の制御の難しさがあります。

たとえ大きなトランザクション・パターンの変化やデータの偏りなどが発生しても、データベース上に競合ポイントが発生しにくい基盤環境をつくり上げ、開発メンバーには本来の業務に直結するロジック・フローの開発に専念させることが必要です。データ競合の問題から解放されることで、開発環境での生産性向上はもちろん、プログラムの品質向上にもつながります。

2.4. 高信頼性基盤設計での課題

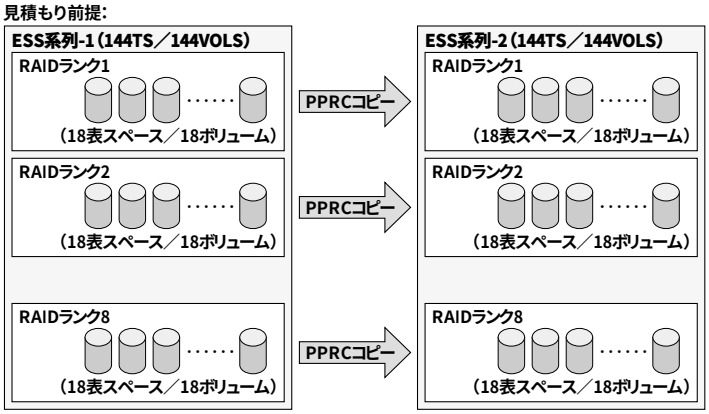
TPFやIMSを基盤とした従来の基幹系システムは、耐障害性の観点でも先行しており、短時間での資源テークオーバーやデータ2重化機能を実現しています。並列シスプレックス構成は99.999%の信頼性を持つといわれますが、例えば銀行勘定系に適用する場合、片系障害発生時には多くの銀行ATM（Automatic Teller Machine：現金自動入金・自動支払機）のタイムアウト時間が1分に設定されていることを考慮し、少なくとも1分以内にテークオーバーすることが求められます。この点に関して、過去のDB2による大規模高信頼性システム構築には二つの大きな課題がありました。特定DB2メンバー障害時のリテインド・ロックの問題と、DASD系列障害時の資源テークオーバー時間の問題です。これらはハードウェア資源を2重化しただけの単純なアプローチでは解決できません。

• メンバー障害時のリテインド・ロックの影響範囲

並列シスプレックス環境でデータ共用を実施している場合、複数DB2メンバー間でのデータ整合性を維持するために、グローバル物理ロック（以下、Pロック）と論理ロック（以下、Lロック）が使用されます。1メンバーで障害が発生した場合、障害発生メンバーの保持していたロックは、リテインド・ロックとしてDB2再始動

図4. 表スペース全体に排他制御が保持されるタイミング

■IBM-ESS環境でのPPRC使用時の最短回復時間（ESS系列障害モデルケース:一部実績値を使用）
見積もり前提:



→DB2バッファー・プール総容量を1Gバイト、更新済みページ最大容量を500Mバイト
ユーザー・データベース1系列当たりの書き出し未了ページ容量は125Mバイトと想定。
→系列障害時は障害TS数は144TSでテークオーバー時間は約28分と推定

#	回復手順	予想所要時間
①	障害TSの停止	5分
②	PPRC対象VOL切り替え	17分
③	障害TSのLPL状態解消（START）	6分
計		28分

②の内訳（一部実績値）

PPRCプライマリー側VOLオフライン:	5分
制御プログラムでの状況確認:	1分
PPRC回復コマンド発行:	5分
PPRCセカンダリー側VOLオンライン:	5分
制御プログラムでの状況確認:	1分

③の前提と内訳
前提：DASDに書き出し未了のページ（LPL状態で回復が必要）は31,250ページ（125Mバイト）とすると1TS当たりの回復ページは218ページ（0.8Mバイト）と想定。
下表は1TS当たりの回復時間

#	DB2内部処理概要	見積り所要時間	計算式
1.	VSAMオープン	0.5秒	経験値
2.	DB2ログ読み込み時間	7.5秒	2チェック・ポイント分（1,000,000ログ・レコード）のREAD。1ログ・レコード＝1Kバイトと仮定。1Kバイト×1,000,000＝100Mバイト＝25KのVSAN-CI 25K×0.3ms＝7.5秒（0.3msは1CIの読み込み時間）
3.	LPL対象ページのREAD	2秒	1ページ当たりのREAD時間を10msと仮定 10ms×218ページ＝約2秒
4.	DB2ログ適用時間	2秒	1ページ当たりのREAD時間を10msと仮定 10ms×218ページ＝約2秒

→障害TS数は列系障害で144。これを2個の-STARTコマンドで多重回復。
単一の-STARTコマンド（72TS回復分）での予想所要時間
＝7.5秒（ログ読み込み）＋［0.5秒（VSAMオープン）＋2秒（LPLページREAD）＋2秒（ログ適用）］×72
＝331.5秒＝約6分（多重化コストやDISPLAYコマンドでの確認操作を考慮）

図5. DASD2重化環境での予想テークオーバー時間

完了まで他メンバーからのロック対象資源へのアクセスが制約されますが、通常、影響を受けるのはあくまで特定ページあるいは特定行と認識されています。しかし特定のタイミングでメンバーが障害を起こすと、DB2カタログ・ディレクトリーを含む特定表スペース全体に排他ロック（以下、Xロック）が掛かり、実質的に他メンバーのデータベース処理が不可能な場合があります（図4）。リテインド・ロックの早期解消には、障害DB2メンバーの高速再始動が必要ですが、再始動まで他メンバーの処理継続を大きく阻害する表スペース全体のリテインド・ロック状態を抑制する方策が必要です。

• DASD系列障害での対応方法の選択

DASD系列はRAID（Redundant Arrays of Independent Disks）構造により特定トラックやDA（Device Adaptor）などの同一系列内のハードウェア機構は2重化されています。しかし系列全体の障害を想定した場合、データの2重化メカニズムなしでは数時間～数十時間のシステム停止を覚悟する必要があります。銀行勘定系を想定した大規模基幹系構築では、長時間停止を防ぐために何らかのデータベース2重化方式の適用が必要です。しかし通常のESS（Enterprise Storage Server）環境でのPPRC（Peer-to-Peer Remote Copy：対等遠隔コピー）などの単純な2重コピー方式では、最短の見積もりでもテークオーバーに30分近くを要し、高信頼性システムとしての基本要件を満たせません（図5）。テークオーバー時間は前述のように少なくとも1分以下に抑える必要があり、新たなアプローチが必要になります。

■CPUキャパシティー予測例：典型的な処理パターンを設定し予測する。
原単位グループ1. データ転送要素（MQ関連）
TCPIP RECEIVE＝10K MQPUT（データ・キュー）＝32K（5Kバイト・パーシステントでコミット間隔＝100）
MQCMIT＝9K
原単位グループ2. 更新タスク（MQ、CICS、DB2、適用業務プログラム）
CICS MQGET（データ・キュー）＝27K（5Kバイト・パーシステントでコミット間隔＝1）
MQCMT＝9K　SQLCMT＝50K　SELECT＝50K　UPDATE＝56K　INSERT＝70K
CICS＋アプリケーション＝40K（DB2更新アプリケーション構造に依存）

処理パターン1（24%）：SQL　70K（INSERT×1）
処理パターン2（16%）：SQL 270K（SELECT×2＋UPDATE×2＋INSERT×1）　→ 上位約20%の処理パターンが対象
処理パターンn　（4%）：SQL 100K（SELECT×1＋UPDATE×1）

・パターン1系メッセージ・プロセス当たりの処理コスト→TCPIP＋MQPUT＋MQCMIT／100＋MQGET＋（処理パターン1 SQL）＋（CICS＋アプリケーション）＋（MQCMIT＋SQLCMT）／50＝180K
・パターンn系メッセージ・プロセス当たりの処理コスト→TCPIP＋MQPUT＋MQCMIT／100＋MQGET＋（処理パターンn SQL）＋（CICS＋アプリケーション）＋（MQCMIT＋SQLCMT）／50＝210K

各SQLコストは範囲の上限を想定。通常は上記コストの半分程度
→MQMのバッチ・サイズ＝100と想定
→50メッセージ単位にCICS同期点処理。ショート・コミット処理でのCPUコストおよびDB2ロギング・コスト増加を回避

■トランザクション処理プロセッサー・コスト算出

1. トランザクション処理コスト
（メッセージ比率でパターン1～n系の命令数の加重平均）×ピーク・メッセージ量×1.3（安全率30%）÷1,000,000＝必要MIPS値
計算例：220,000（220K）命令×1,200メッセージ／秒×1.3÷1,000,000（命令／秒）＝343.2MIPS
2. SCPコストとして＋10%
3. RLL方式の追加コスト＋3%（非共用）
4. データベース共用時の追加コスト＋25%（共用データベース比率分）
5. データベース共用時のRLL方式の追加コスト＋10%

図6. 原単位からのCPUキャパシティー予測モデル

■I/Oキャパシティー＆CICSタスク数予測例：
見積もりの前提：DASD環境 ESS F20
更新メッセージ・レート＝1,200件／秒
トランザクション・プロファイル SSELECT×2、UPDATE×1、INSERT×1ですべて索引経由のランダム処理。

•必要入出力数＝4（SELECT数+UPDATE数+INSERT数）×2（リーフ・ページI/O+データ・ページI/O）＝8
•総入出力数（1秒間）＝8×1,200＝9,600 → ESS最低2系列必要
•処理時間（平均タスク滞留時間）＝必要入出力数×10ms＝80ms（10msはESS 平均想定I/O時間）
単一タスクでの1秒間のトランザクション処理数＝1秒÷平均タスク滞留時間＝1,000ms÷80ms＝12.5
必要CICS更新タスク数 ＝転送トランザクション・レート÷単一タスクでの1秒間のトランザクション処理数
＝1,200÷12.5＝96 → 96個のCICS更新タスクが最低必要

補足1：入出力数の計算で索引階層に関係なくルート・ページとノンリーフ・ページはバッファー・ヒット率＝100%と仮定。現実にI/O発生の可能性は非常に低い。
リーフ・ページおよびデータページについてはバッファー・ヒット率＝0%として見積もり。現実にはヒット率は40～98%を変動（負荷余裕分として組み入れ）。
補足2：処理時間で1回当たりのI/O時間を10ミリ秒としているが現実的環境でのI/O時間は4～8ms程度。
（これも上記同様に負荷余裕分として組み入れ）

図7. 正確なキャパシティー予測（I/O資源&常駐タスク数）

3. DB2超大規模基幹系構築アプローチ

3.1. 正確な原単位予測と限界能力の把握

前章で述べたように、DB2環境ではデータベース構造や処理フローの自由度が高く、またデータベース・アクセスを行うSQL操作においてもシステム・コストに直結するアクセス経路を直接意識しないため、従来の基幹系に比べて慎重なキャパシティー予測作業が必要です。できるだけ早い段階で、業務要件に基づき、代表的なトランザクション処理パターンと、データベース・プロファイル、システム運用体系を確定させることが重要です。

3.1.1. ハードウェア資源（CPU、I/O）

• CPU資源

CPU資源予測の最初の段階は、各処理要素の原単位の把握です。図6に基本的なキャパシティー予測の方法を示します。お客様業務要件を正確に認識し、代表的な処理フローを選択することが重要です。特に基幹系の場合は、トランザクション発生量の上位20%程度で負荷全体の80%を占めるといういわゆる“80/20の法則”に基づき、上位処理パターン・グループをある程度選別した後に各パターン負荷の原単位を計算し、加重平均して全体のCPUコストを予測します。また図で最もコストの高いSQL操作の原単位は、指定列数や列属性（文字・2進・10進）、ホスト変数などで変動するため、データベース・プ

ロファイルも早い段階で明確にしておきます。

さらに基幹系での並列シスプレックス環境では、基本的にすべてのデータベースがデータ共用状態（すなわちクローン・システム）に設定すること、データベース共用時の追加コストとして＋25%程度含めなければなりません。この＋25%という数字はzSeries™でCF（Coupling Facility：結合機構）接続に1Mバイト/秒のスピードで転送可能なICB3（Integrated Cluster Bus：高速結合機構）かIC（Internal Channel：内部チャネル）を使用し、大規模更新トランザクション環境で全データベースを共用したときの実績値であり、それ以外の環境でのコストは変動します。またデータ競合最小化のためにさらにRLL（Row-level Locking：行単位ロック）適用コスト＋10%が発生する場合があります。

• I/Oおよび処理タスク数

I/Oコストおよび大量のトランザクション処理に必要なCICS常駐タスク数も、CPUと同様に予測できます。図7に予測例を示します。使用DASDタイプ、更新メッセージ量、トランザクション・プロファイルを仮定しますが、もちろん図の予測はトランザクション部分しか含まれておらず、センター・カットなどの大規模バッチやバックアップ・再編成などのユーティリティー運用に伴うI/O操作との競合なども十分に考慮する必要があります。

当然、DASDの限界能力を把握する一環として、他社製DASD製品やボリューム・アドレス数の制約などにより、PAV（Parallel Access Volume）機能が適用できない場合についても配慮が必要です。

3.1.2. ボトルネック要素（仮想記憶域、DB2ログ）

CPUやI/O資源とは異なり、仮想記憶域やDB2ログなどのボトルネック要素で問題が発生した場合には、システム基盤構成を見直す必要が生じるため、慎重に予測します。

• 仮想記憶域

大規模基幹系の仮想記憶域不足でDB2システム障害の原因となる主なエリアは、ECSA/CSAとDB2－データベース管理アドレス空間（通称DSNDBM1）内の16Mバイト・ライン以下の仮想記憶域です（図8）。

特にPC（Program Control：プログラム制御）＝NO（ロック制御ブロックをECSAに保持）を指定している場合、総ロック制御ブロック数増加によりESCAでストレージ不足が発生すると、IRLM（Internal Resource Lock Manager）の異常終了に伴いDB2メンバーがダウンします。さらにデータベース共用環境では、他DB2メンバーのIRLMはリテインド・ロック制御のため、異常終了したメンバーのロック情報を別システムのECSA内に追加で保持する必要があり、同様なECSA不足が連鎖的に発生し、結果として全DB2メンバーがダウンする可能性があります。この連鎖ダウンのリスクを回避するために、最大ロック数の制御はもちろんですが、データベース共用環境ではPC＝YES指定により最大サイズに制約があるECSAでなく、IRLMアドレス空間上にロック制御ブロックを保持することでECSAストレージが不足しないように細心の注意を払います。最近ではロック・アボイダンス機能（ロック要求の絶対量を減らすメカニズム）によりPC＝YES指定でのクロス・メモリー操作に伴うCPUコストの増加は最大でも1～2%程度です。

もう一つの注意要素は、DSNDBM1の16Mバイト・ライン以下のエリアであり、主に作業域とVSAM（Virtual Storage Access Method：仮想記憶アクセス方式）制御ブロックにより使用されます。VSAM制御ブロックに関してはオープン・クローズ・コスト抑止のために大きなDSMAX値を指定しているとストレージ不足によりDSNDBM1が異常終了する可能性があります。通常は、最大オープンVSAM数が20,000以上の環境でもDSMAX値は7,000以下程度に抑えます。高信頼性システムではオープン/クローズのコスト低減よりも、異常終了を最小限に抑制することが重要と考えます。作業域についても最大スレッド数に上限を設け、VSAM制御ブロックの場合と同様にエリア不足にならないよう注意します。

大規模システムではSWA（Scheduler Work Area：スケジューラー作業域）＝ABOVE指定が必要ですが、IMSと同居する環境ではTIOT（Task I/O Table：タスク入出力テーブル）などの制御ブロックを直接参照しているプログラムが存在し、ABOVE指定ができない場合があるので注意が必要です。

• DB2ログ競合

DB2に限らず大量トランザクション処理や大規模バッチ更新環境で、常に問題となるログ書き出し能力では、DASD能力の向上で飛躍的に上がりましたが（図9）、これはログ同期

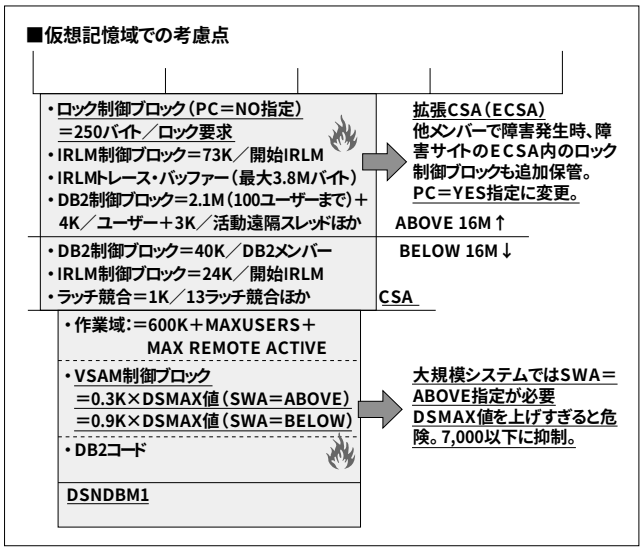


図8. 仮想記憶域上に制約（ECSA/CSA&DSNDBM1）

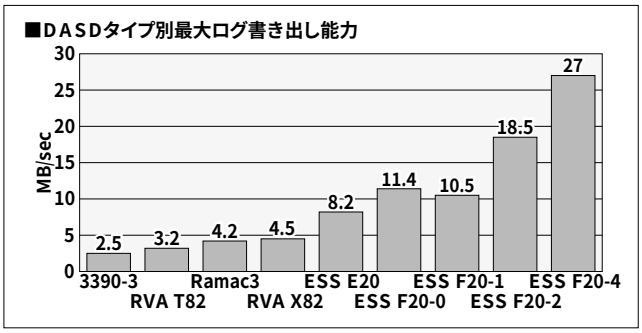


図9. 最大DB2ログ書き出し能力

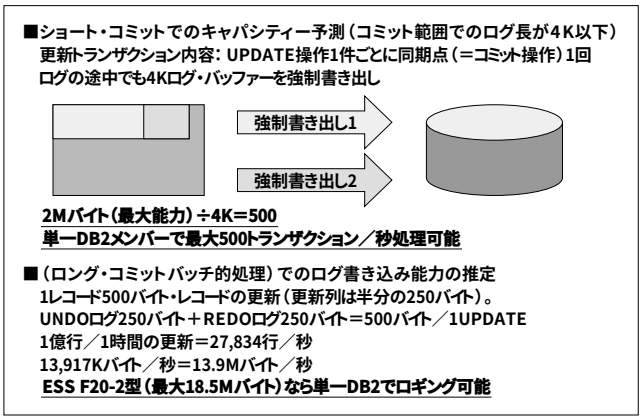


図10．DB2ログ・キャパシティー予測

書き出し処理を伴うロング・コミット操作（コミット発生頻度が少ないバッチ処理など）で適用可能な数値です。しかしトランザクション1件ごとにコミット処理が発生するようなショート・コミット操作の場合は、ログ・バッファー内に一部しか更新ログが含まれていなくても、ログ・データ・セット内のVSAM-CI（Control-Interval：制御インターバル）上の同一個所に強制書き出しされます。結果として同一ログ・バッファーの内容が何度も書き出され、処理効率は低下します。ショート・コミットの場合の書き出し能力は、過去の実測値ベースで最大2Mバイト程度です。ショート・コミット操作とロング・コミット操作で予測手法は異なります（図10）。

3.2. 統計値の反映と変更リスク回避

2.2節で論述したように、正確な統計値の反映によりアクセス選択環境を整えるのは非常に重要ですが、通常オプションでのRUNSTATSやBIND操作では対応できない問題が存在します。第1にデータの偏りを正確に把握する統計データの深さと、SQL上のWHERE条件節での変動要素であるホスト変数対応が挙げられます。データの偏りとホスト変数の内容を反映しないと、より正確なアクセス経路は選択されません。第2にサービス・イン当初の空データ状態にどう対応するかです。0件状態の統計情報の下で大量のデータがINSERTされた場合、次回のRUNSTATS＋REBINDまでは表スペース・スキャン選択によるパフォーマンス低下と、大量のデッドロック発生のリスクを抱えます。第3に、基幹系の場合に最も重要なのが、RUNSTATS＋REBIND後にアクセス経路が変更され、結果的に生じる可能性のあるパフォーマンス低下のリスクをどう回避するかです。

図11に要約したように、処理効率と信頼性両面が要求される基幹系システムでは、第1の問題に対してはRUNSTATSでデータ突出値反映指定（NUM、COUNT）、列相関統計指定（KEYCARDS）とBIND時のREOPT（VARS）によるホスト変数対応を、すべての表スペースおよびパッケージに対し検証します。第2の問題についてはマニュアルによるカタログ更新で次回のRUNSTATS時期までに挿入予定データの数倍規模の擬似統計値をセットします。基本的に統計データ上の行件数が現実にINSERTされた行数よりも多ければ、カタログ統計値と現実のデータの間でギャップによる問題は報告されていません。これは、件数が多いほど索引使用が選択されやすくなり、各種臨界値（RIDソートなど）に達しないなど、さまざまな理由が挙げられます。逆に0件RUNSTATS後の大量データINSERTのように統計値が現実のデータ件数よりも小さい場合、種々の問題が発生することを認識しておくべきです。第3の最も重要なアクセス経路変更に伴うパフォーマンスの低下リスクの回避には次のように対応します。

- 基幹系では基本的に静的SQLが100%近くを占めることを考慮し、RUNSTATS後のBIND時にアクセス変更も検知する手段を確立します。

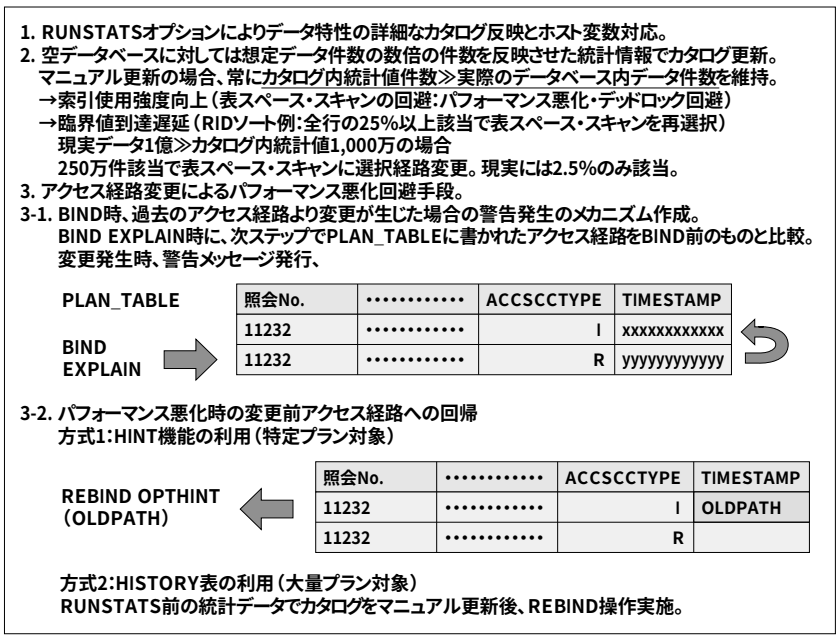


図11. RUNSTATS運用とアクセス経路変更リスク

- パフォーマンス低下が発生した場合に、変更前のアクセス経路への自動回帰システムを準備します。

最初に確定したアクセス経路を変更させないために、最初のRUNSTATSのみとし、以降はカタログ統計を更新しないアプローチ方法ももちろん存在します。しかし、データ特性の変化が激しい場合には定期的な再編成＋RUNSTATSの実行が推奨されます。ただし、上記の回避手段が実現できれば、問題発生の事前検知や短時間での戻し作業が可能になります。

3.3. 競合フリーなデータベース設計

2.3節でも述べたように、事前予測に基づいてデータ分散やトランザクション処理でのキー分散を実施しても、大量トランザクション環境ではサービス・イン後の業務環境の変化により、データベース・ホット・ポイントが発生する可能性が高くなります。いったん発生すると、問題回避には処理フローやデータベース構造の変更が必要になり、短期的な対応が困難です。回避するために以下の3要素について検討を加えていきます。

- データ競合が起こりにくいシステム基盤環境の提供。
- 業務処理フローでの標準的な参照／更新順序の順守（ABC順の表更新、キー順更新など）。
- アクセスが集中しやすい時系列表・採番表などへの配慮。

3.3.1. デッドロック・タイムアウトの抑制

1,000件/秒を超える大量更新トランザクション環境で、現実に適用が検討されているデータベース構造には、タスク区分対応方式とキー順序化RLL（Row Level Lock：行単位ロック）方式があります（図12）。

タスク区分対応方式は、特定表・特定区分の更新を特定タスクでしか許さない方式であり、基幹系から情報系などへの大量のディレード更新処理などに使用されます。区分とタスクが1対1の関係であり、データベース内のデータが複数タスクから決して更新されないため、タスク間でのデータ競合は発生しません。この方式はデータ共用コストを最小限に抑制しますが、問題は、単一のタスクが複数の表や区分にわたって更新する場合に適用できないことと、結果的に特定区分は単一DB2メンバー側からしか更新できない非クローン構造となり、DB2メンバーの障害時に他DB2メンバーから即時にアクセスができない点です。

一方、キー順序化RLL方式は、同期点内で更新キーをいったんソートしてキー順にRLL環境で各行を更新するため、デッドロックは発生しません。この方式の長所は、単一タスクで複数区分の更新が可能であり、並列シスプレックスの最大の特徴であるクローン構造（共用データベースが複数のメンバーから常時アクセス可能）を十分に生かせることです。ただし3.1.1項で述べたようにデータ共用コスト＋RLL追加コストが必要なことと、同期点内で処理順序の逆転が発生するため最新データのみを保存するマスター更新などではユーザー・プログラムによるタイム・スタンプ制御が必要になる場合があります。キー順RLL方式と通常ページ・ロック方式との差を図13に示します。

基幹系での設計アプローチは、通常、共用コスト削減よりもクローン構造による信頼性が重視されることと、単一タスクから複数表・区分への更新制約は基幹系業務では不適切な場合が多いことなどから、データ競合を最小化する基盤構造としてはキー順RLL方式が一般的です。このシステム構造下での開発では、データ競合が大幅に抑制されるため適用業務設計やデータベース設計が容易になり、結果として開発生産性向上やシステム全体の環境変化に対する強度が飛躍的に増加します。2方式のそれぞれの長所／短所を表1にまとめておきます。

3.3.2. コミット操作と採番システム

通常、大規模更新トランザクション環境においては、ログの負担軽減や処理効率を重視するため、複数の同期点処理をまとめて単一の同期点で処理する場合が多くあります。しかしこのアプローチは、図14に示すようにデッドロックなどのデータ競合が発生しやすいことを認識する必要があります。たとえ

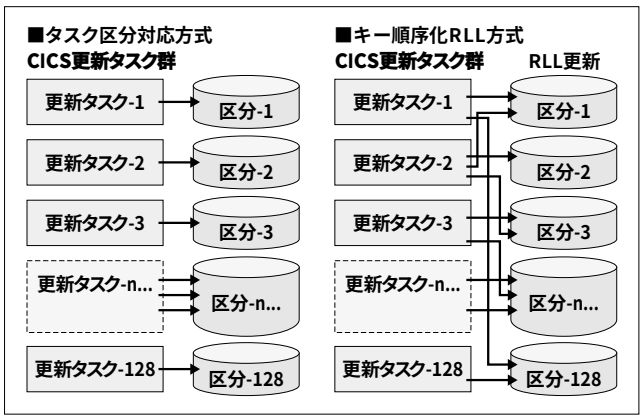


図12. データ競合回避基盤

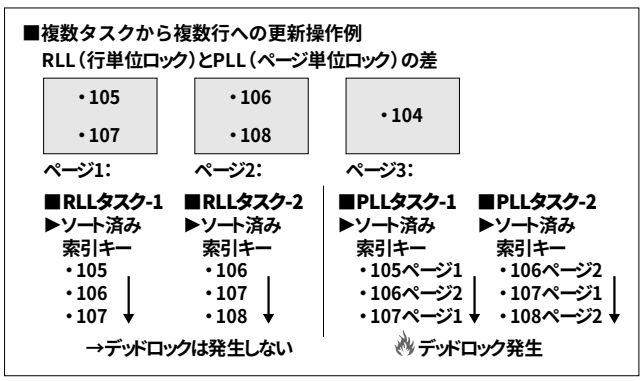


図13. キー順序化RLL方式の特徴

1UPDATE操作ごとに同期点を設定しても、3.1.2節で述べたように、単一DB2メンバー当たり最大500件/秒程度までの同期点を処理できることを考慮すれば、単一同期点にまとめる操作は避ける方がデータ競合上は適切といえます。

次に受注番号発行などを行う採番処理では図15のように、特にデータベース共用環境ではページPロック競合によるネゴシエーション操作や、データベース共用環境でのロック保持側の要求を優先処理するため、WAIT側でのタイムアウト発生などで、採番表とアクセスするトランザクションは量的に非常に厳しい制約を受けます。そのため大規模基幹系ではトランザクションの採番対象として、全社採番ではなく、特定支店や部門ごとのグループ採番を採用するのが一般的です。

3.4. 高信頼システム構築アプローチ

3.4.1. 並列シスプレックス高速テークオーバー

並列シスプレックス環境では、プロセッサー、CF、チャネル系列などのハードウェア環境のすべてを完全2重化が可能です。しかし2.4節で述べたように、DB2再始動までのリテインド・ロックの影響を最小限に抑えるには、表スペース単位全体の競合を防ぐことがまず必要です。すなわち図4に示したPロックとLロックのおのおのの矢印が示す排他期間をどう抑制するかということです。

カタログ・ディレクトリー部分に関しては、DB2初期パラメーターDSNZPARMでのSPRMHAVL指定をオンにすることでPロックを常にIX（Intent eXclusive：排他意図）状態に置くことができます。ユーザー表スペースについては、図14の①の初期フェーズでは、スタート時にメンバーA/Bから重要な共用対象データベースに対して同時に擬似的な更新を実施し、表スペースPロックのXモード化を防止します。

また⑦のトリガーを消すために、重要なデータベースに関しては一定期間ごとに参照を実施するのが現実的な方法です。もちろんSPRMHAVLのユーザー・データベースへの適用拡大は、現在必要な機能として求められています。※1 次に、⑥の状態のLロックを防ぐための解決策は現状では存在しないものの、※2 ページLロック波及の中断操作がPロックのモードにリンクできれば（すなわちPロックがIXモードの間は、子ロックの書き出しを実施）、Pロックと同様のアプローチで解決でき、現在その機能が求められています。もちろんこれらはデータベース共用コストが増大するのは承知の上でのアプローチ方法です。

表1. データ競合最小化方式の比較

基本設計方式	タスク―区分対応方式		キー順序化RLL方式
区分キー対象	事前予測（店群分割）	ハッシング・キー分割	
タスク・ワークロード・バランス	△事前予測精度次第	○ハッシングにより処理対象キーは均一に分散	◎処理状況に応じ動的に負荷分散
クローン構造	△片系障害時資源の1/2がアクセス不可能	△同左	◎障害時、ほかの全DB2メンバーより実行可能
処理コスト	◎ページ・ロック、非共用なので最小限	◎同左	△RLLロック・コスト（＋2～3%）必要特にデータ共用時は要注意
照会の容易さ	○そのままで対応可能	△ハッシング・キー除去およびマージ・再分割が必要	○そのままで対応可能
業務上の制約	△1タスクで複数区分の更新ができない	△1タスクで複数区分の更新ができない	△更新順序逆転の対応 主キーが原則的にユニーク
タスク数変更時の影響	△データベース区分数の変更が必要	△同左	○タスク数とデータベース区分数は独立

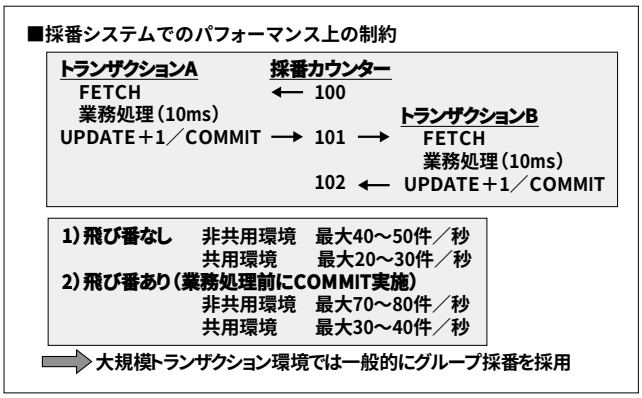


図15. 採番システムでの制約

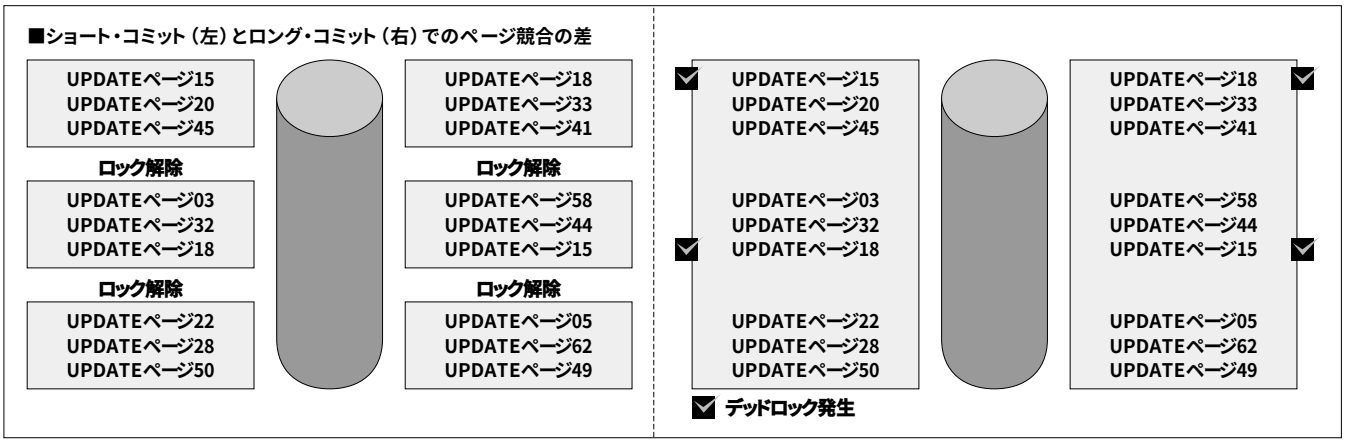


図14. ショート・コミットとロング・コミットの差

第2に、DB2高速再始動については、図5に示したようにDB2再始動のために最大2チェックポイント・ログ100Mバイトを読むのにかかる時間が7.5秒程度であり、バッファー・プールなどのGETMAIN操作を省いたDB2クイック再始動は経験値で30～40秒程度で完了し、自動再始動メカニズムを併せて1分以内のリテインド・ロックの解消が可能となっています。

またCICSとの間のINDOUBTスレッド解消に必要なCICS-AOR（CICS - Application Owing Region）の再始動時間はプログラム資源の自動インストールやトランザクション資源定義の絞り込みなどで60秒以内の再始動時間を実現させます。ただし、これらの再始動時間のさらなる短縮については、DB2が継続的に取り組むべき課題です。

※1：2003年3月に使用可能となったAPAR：PQ69741により、表スペース定義時にCLOSE（NO）と指定することで、⑦のPCLOSE-T経過後でもP-LOCKがIX-Xにアップグレードされることを防ぐことが可能になりました。

※2：2003年1月発表されたDB2第8版において、Lロックに関する共用メカニズムが改善され、リテインド・ロックとして波及することはなくなりました。そのため、上記Lロックに関する考慮は不要になります。

3.4.2. データベース2重化アプローチ

2.4節のDASD系列障害での対応方法の選択で述べたように、データベース2重化にPPRCを使用した従来のアプローチ方式では、DASD資源上で系列障害が発生した場合、ボリュームのオフライン・オンライン操作やコマンド発行などで最低限でも30分程度のテークオーバー時間が必要となります（図5参照）。

最新機能のGDPS（Geographically Dispersed Parallel Sysplex）の一環として実現されたハイパースワップ機能を従来のPPRC機能と組み合わせれば、非常に短時間でのDASD系列障害からのテークオーバーが可能です（図16）しかしさらに高いレベルのテークオーバー・システムを実現するには、以下の課題点が残ると考えます。

- ハイパースワップはソフトウェア（OS/390®-IOS：IOスーパーバイザーとGDPS）とマイクロコードにより実現されていますが、ESS環境が前提であり、それ以外のDASD環境の基幹系システムには適用されません。すなわちハードウェア環境からの独立性を高めることが今後の課題となります。
- 系列障害は、通常、マイクロコードを含むハードウェアの問題で発生します。マイクロコード依存のテークオーバー方式以外の新たなアプローチ方式も必要です。

結論として、ハイパースワップのアプローチ方式はデータベース2重化を実現する上で非常に効果的ですが、本来であればIMSのMADS（Multiple Area Data Set：複数エリア・データ・セット）と同様に、SCP本体あるいはDB2による完全なソフトウェアによる2重化機能についても、お客様の選択肢とSCPや装置からの独立性を高める意味で早期の実現が望まれます。

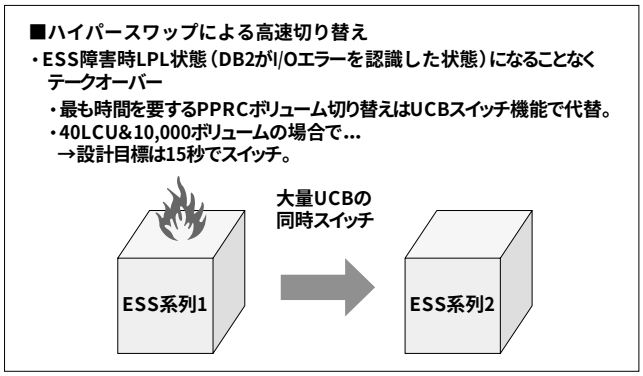


図16. ハイパースワップによる高速資源テークオーバー

4. おわりに

DB2による銀行の勘定系構築や、1,000件/秒を超す高速トランザクション処理システムの実現は、開発生産性向上やオープン・システムへの対応という面を含め、多くのお客様より望まれてきた課題です。

都市銀行勘定系に必要な高信頼性と、座席予約システム実現に必要な高速大量トランザクション処理、そしてTバイト級のデータベース環境への対応など、DB2システムはより多くの課題を克服してきました。今回の考察でも幾つかの項目で一部課題が残るものの、今後十分に実現可能と考えます。DB2では敷居が高いとされてきた領域に今後チャレンジしていくエンジニアにとって、本論文で検証してきたDB2大規模基幹系の設計アプローチが少しでも役立てば幸いです。

（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] John J Campbell, 'Virtual Storage Management Before and After zSeries'.
[2] Akira Shibamiya, 'DB2 for z/OS and OS/390 Performance Update', IDUG 2001 Conference