Vol. 33 No. 12　　情　報　処　理　　Dec. 1992

解　説

データベースプロセッサ

NonStop SQL のアーキテクチャ†

樋川英治††　渡辺榮一†††

1. はじめに

タンデムコンピューターズの NonStop SQL[1]（以下 "NS—SQL" と略す）は，オンライントランザクション処理（OLTP）のために開発された関係データベース管理システム（RDBMS）である．従来の RDBMS は，意志決定支援などの非定型業務を主な用途としているが，NS—SQL は，OLTP を典型とする定型業務に使用できる機能と性能，信頼性，処理能力の拡張性を備えている[2]．

以下では，NS—SQL の設計概念，システムアーキテクチャ，並列処理方式，QUERY の最適化，性能評価について述べる．

2. 設計概念

2.1 設計目標

NS—SQL の設計目標は，次のとおりである．

1) フォールトトレラントなデータアクセス

データアクセスの途中でハードウェアの一部が故障しても，目的とするデータをアクセスできる．

2) モジュールによる拡張性

並列アーキテクチャに基づき，数十台のプロセッサを1システムとして構成し，毎秒数百件の規模のトランザクション処理を可能にする．

3) データと実行の分散

構内網（LAN）および広域網（WAN）の環境で，データの分散と実行の分散を可能にする．

4) システムの統合

ネットワーキングおよびトランザクション処理ソフトウェアを統合した RDBMS を実現する．

2.2 実現の基準

NS—SQL は，RDBMS の先駆的事例である System R[3]，SQL/DS，DB 2[4]，および米国標準 ANSI SQL[5] の定義に準拠しつつ，分散データの支援，高性能化，オペレータインタフェース，システムソフトウェアとの統合などに独自の特徴をもつものとなっている．

3. アーキテクチャ

3.1 並列アーキテクチャ

NonStop システムは，並列ハードウェアアーキテクチャを採用している（図-1）．それを支援するために，プロセスとメッセージの概念に基づくソフトウェアアーキテクチャが採用されている．プロセスは，機能・故障・拡張の単位となるソフトウェアモジュールであり，メッセージを介して互いに通信しあう．共有メモリは，使われていない．その理由は，故障の封じ込めの点で劣るだけでなく，プロセスがネットワーク内のどこにでも存在できる能力を制限するためである．

3.2 NS—SQL とフォールトトレランス[6]

NonStop システムは，ハードウェア，ソフトウェアに冗長度をもたせることにより，フォールトトレランスを実現する．ソフトウェアは，プライマリプロセスおよびバックアッププロセスにより二重化される．ディスクサーバ（後述）の二重化と，トランザクション保護のための DO—UNDO—REDO メカニズムにより，NS—SQL は，プロセッサや媒体の単一故障に耐える．すなわち，NS—SQL のフォールトトレランスは，プロセスの二重化とトランザクション機構の組合せに基づいて実現されている．

3.3 NS—SQL の構造と動作の概要

NonStop システムは，メッセージを介して通信しあう疎結合マルチプロセッサであり，プロセス間通信のオーバヘッドが性能のボトルネックになりうる．この問題を避けるために，NS—SQL は，

† Architecture of NonStop SQL by Eiji HIKAWA (Tandem Computers) and Eiichi WATANABE (Info Strategies).
†† 日本タンデムコンピューターズ
††† インフォストラテジ

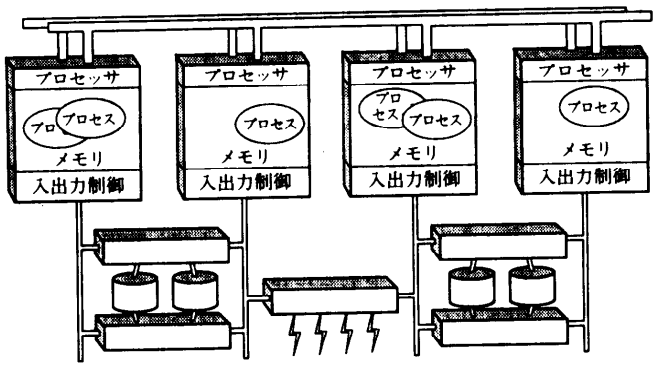


図-1 NonStop システムの並列アーキテクチャ

"OS の上" に位置するのではなく，データ管理に関する OS の機能と統合して実現されている[1]．

1) データアクセス

NS—SQL では，ユーザは，COBOL 85 と SQL で書かれたプログラムまたは会話型インタフェースから，データをアクセスする．ユーザプログラムは，PATHWAY[7] と呼ばれる DB/DC システムのサーバとして動作する．ユーザプログラム中の SQL 文は，SQL エグゼキュータを起動する．エグゼキュータは，アプリケーション環境の中で走るライブラリルーチンであり，ファイルシステムを経由してさまざまなテーブルからデータを収集し，組み合わせ，ホスト言語（ユーザプログラム）の変数に結果を入れて返す．

2) ファイルシステム

ファイルシステムは，ファイルとインデックスのオープン，パーティションの管理，アクセス要求を適切なディスクサーバへ振り分けること，応答の生成などを行う．

3) ディスクサーバ

二重化されたディスクボリュームは，ディスクサーバによって管理される．ディスクサーバは，ディスク空間，アクセス経路，ロック，ログレコード，主記憶のバッファプールの管理などを行う．ディスクサーバは，データベースをアクセスして述語（検索条件）を満たすデータを見つけ，ユーザが要求した SQL の演算操作を行う．

4) コンパイラ

NS—SQL コンパイラは，分散 QUERY コンパイラ技術を使用し，入出力コスト，CPU コスト，メッセージコストを組み合わせた費用関数を最小にする実行プランを決定する．コンパイラは，機械語のコードを生成する代わりに，SQL エグゼキュータによって解釈実行される制御ブロックを生成する．

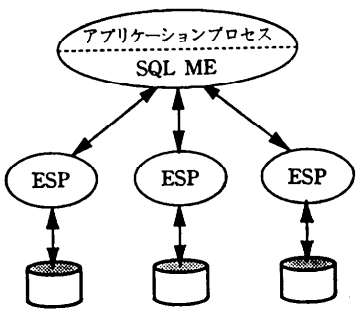
4. 並列処理方式[8]

4.1 QUERY の並列処理

NS—SQL における QUERY の並列処理は，複数のエグゼキュータサーバプロセス (ESP) を並列に実行することによって実現される (図-2)．ESP は，マスタエグゼキュータ (ME) によって監視される．各 ESP は，単一のパーティションに対する QUERY を受け持ち，結果を ME に返す．ME は，その結果をユーザに渡す．NS—SQL では，選択・更新・挿入・結合などの演算操作を並列に処理することができるが，ユーザは，並列処理を実際に行うかどうかを，制御文 CONTROL EXECUTOR PARALLEL EXECUTION {ON|OFF} によって選択することが可能である．

4.2 データの再編成

NS—SQL の並列処理能力を最も効果的に利用するためには，データが適切にパーティション化されていることが必要である．データおよび関連するインデックスは，通常，複数の CPU のディスクに分割して配置されるが，テーブルによっては，そうなっていないこともある．オプティマイザは，データを再編成（再パーティション）すべきかどうかを，コストの評価に基づいて決定する．その決定に従って，ESP は，実行時に，CPU ごとに一時的なパーティションを作成する．ESP は，基本テーブルから選択されたタプルにハッ



ME=Master Executor
ESP=Executor Server Process

図-2 QUERY の並列処理

シュ関数を適用し，該当する一時的パーティションを決定し，タプルをそのパーティションに挿入する．図-3 は，三つの基本テーブルを，あるカラムの最初の文字の範囲 (a-g, h-n, o-z) に従って，三つの一時的なパーティションに再編成する様子を示している．

4.3 インデックスの並列処理

基本テーブルが変更されると，関連するインデックスの変更も必要になる．まず基本テーブルのローが変更される (図-4 のステップ1)．次に，もし複数のインデックスがあれば，それらの変更は，並列的に行われる (図4のステップ2)．

5. QUERY の最適化

5.1 NS—SQL のオプティマイザの概念

NS—SQL のコンパイラには，最も効率的なアクセスプランを自動的に選ぶためのオプティマイザが組み込まれている．オプティマイザは，コンパイル時に，以下の項目について，与えられた QUERY の入出力コスト，CPU コスト，メッセージ通信コストを評価し，コストの最も低いアクセスプランを採用する．

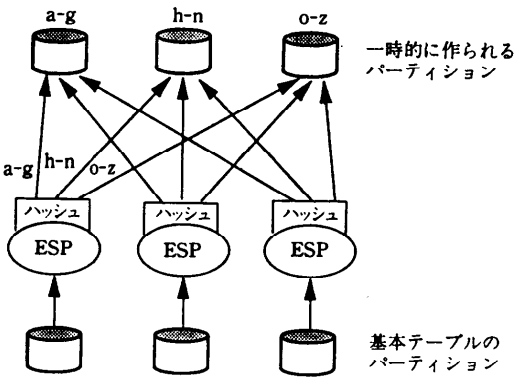


図-3 並列処理のための再編成

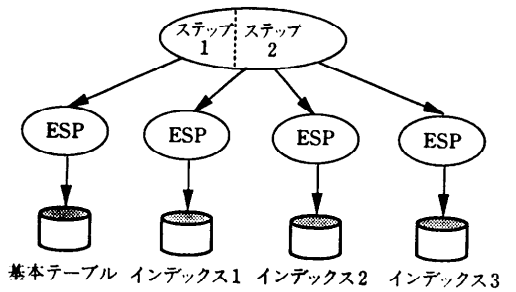


図-4 インデックスの並列処理

1) テーブルとインデックス

テーブルは，物理的には，キー順 (Bツリー)，相対 (直接アクセス) またはエントリ順 (順アクセス) のファイル形式をとり，一次キーをもつとともに，複数の二次インデックスをもつことができる．一次キーによるアクセスよりも，二次インデックスによるアクセスのほうが，物理入出力の回数が減らせる場合には，二次インデックスを使用するアクセスプランが選ばれる．

2) 逐次ブロックバッファリング

NS—SQL は，データ転送を効率的に行うために，ディスクプロセスとユーザプロセスの間で，伝統的なバッファリング (物理逐次ブロックバッファリング＝物理 SBB) とともに，仮想逐次ブロックバッファリング[1] (仮想 SBB) を支援している．NS—SQL に固有の仮想 SBB では，ディスクプロセス自身が選択や射影の演算操作を行うので，メッセージの通信量を一層減らすことができる．

3) セレクティビティ

NS—SQL では，コマンド (UPDATE STATISTICS) で得られた実際の統計情報に基づき，QUERY のセレクティビティを評価する．数値項目のカラムに対するセレクティビティは，(第二最大値－求める値)/(第二最大値－第二最小値) で評価する．また，「カラム＝値」の形式の述語 (検索条件) におけるセレクティビティは，(1/そのカラムのユニークな値の数) で評価する．ここで，統計情報から得られる最大値および最小値については，極端な値を避けるように配慮されている．

4) 結 合

NS—SQL は，二つのテーブルが等号 (=) で結合される "equi-join" と，それ以外の "theta-join[2]" を支援している．また，結合演算の実現技法として，nested loop, sort-merge-join, hash join を支援している．

5) 逐次処理と並列処理

テーブルとインデックスがパーティションとして分割され，UPDATE や DELETE などのアクセスが複数のパーティションにおよぶ場合には，オプティマイザは，並列処理と逐次処理のコストを比較して，最適のプランを選ぶ．

5.2 最適化とユーザの役割[9]

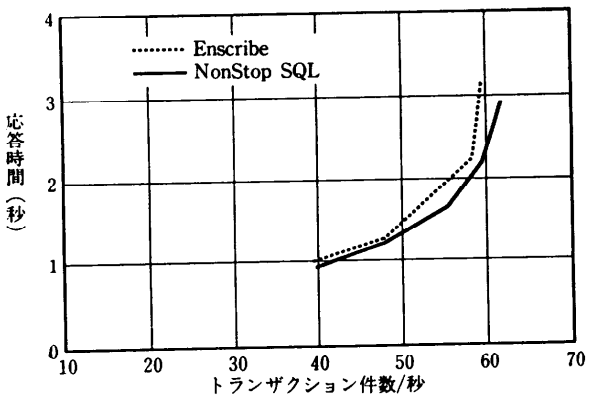
NS—SQL のオプティマイザは，物理 SBB/仮想 SBB，逐次処理/並列処理，結合方法，ソート，サブ QUERY の処理などのコストを評価して，最低コストのアクセスプランを選択する．

ユーザは，QUERY について，より多くの情報を提供し，あるいはアクセスプランについてより多くの選択肢を提供することによって，最適化を高めることができる．たとえば，使用頻度の高いカラムに二次インデックスを追加すること，コマンド (UPDATE STATISTICS) を使ってカラムとテーブルに関する統計情報を常に最新の状態に保つこと，検索条件に複数の値を指定することなどである．これらを通して，NS—SQL の最高の性能を引き出すことができる．

6. 性能評価

6.1 エンスクライブと NS—SQL の性能[1]

伝統的な単一レコードインタフェースのファイルシステムである Tandem のエンスクライブ (Enscribe) と NS—SQL の性能を比較して評価した．ただし，並列処理能力は使われていない．処理のモデルは，デビットクレジットトランザクションであり，三つのテーブルの更新と，一つのテーブルへの挿入からなる．構成は，CPU が 8 台，データベースが合計 11.1 GB である．トランザクションは，9.6 Kbps の回線を通して投入された．図-5 に結果を示す．図から，この処理においては，NS—SQL の性能のほうがいくぶん優れていることが分かる．これは，OLTP においても，RDBMS が実用的性能を提供できることを示している．



(応答時間は 90% のトランザクションに関する値)

図-5 Enscribe と NonStop SQL の性能比較

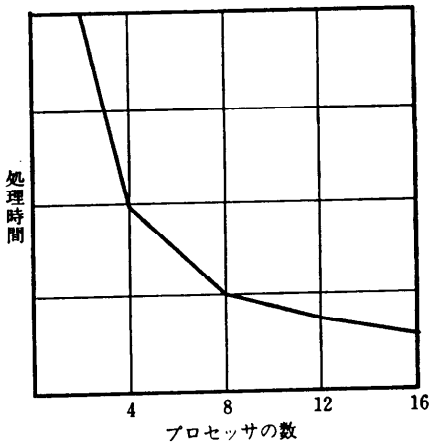


図-6 NonStop SQL の並列処理の性能特性

6.2 NS—SQL の並列処理の性能特性[8]

図-6 は，Tandem VLX システムのプロセッサが 2 台，4 台，8 台，16 台の場合における NS—SQL の並列処理の性能を示している．データベースの大きさは，1.6 GB である．ディスクボリュームおよびパーティションの数は，ここには示されていないが，プロセッサの台数に比例している．それぞれの場合について，データベースの 1% に影響を与える更新処理が行われた．処理に要した時間（経過時間）は，プロセッサの台数にほぼ反比例して，減少していることが分かる．

7. おわりに

NS—SQL は，オペレーティングシステムに統合されており，OS と別個にロードする必要がなく，OS が立ち上がっているときは，常に立ち上がっている．これは，NonStop システムのデータベースマシンとしての一側面を表している．

参考文献

1) The Tandem Database Group : NonStop SQL, A Distributed, High-Performance, High-Availability Implementation of SQL, Tandem Computers (Apr. 1987).
2) Codd, E. F. : The Relational Model for Database Management, Version 2, Addison-Wesley, p. 438 (1990).
3) Astrakan, M. M. et al. : System R : Relational Approach to Database Management, ACM TODS, Vol. 1, No. 2 (June 1976).

4) Date, C. J.: A Guide To DB 2, 2nd ed., Addison-Wesley (1988).
5) Date, C. J.: A Guide to the SQL Standard, Addison-Wesley (1987).
6) 渡辺榮一編訳：フォールト・トレラント・システム，マグロウヒル出版 (1986).
7) 渡辺榮一編訳：OLTP システム，マグロウヒル出版 (1991).
8) Moore, M. and Sodhi., A.: Parallelism in Non-Stop SQL Release 2, Tandem System Review, pp. 36-51, Tandem Computers (Oct. 1990).
9) Pong, M.: NonStop SQL Optimizer: Ouery Optimization and User Influence, Tandem System Review, pp. 22-38, Tandem Computers (Oct. 1990).

(平成4年6月1日受付)

鶴川 英治

1948 年生. 1972 年立教大学理学部卒業. 同年日本ユニバック（現・日本ユニシス）に入社. 金融を中心としたオンラインシステムの設計，開発に従事. 1983年日本タンデムコンピューターズに入社. ソフトウェア教育を担当. 現在，同社教育センター所長.

渡辺 榮一（正会員）

1943年生. 1966年東京都立大学理学部物理学科卒業. 同年日本 NCR に入社. 以来，オンラインシステムの研究開発に従事. 1981年日本タンデムコンピューターズに入社. 1986年米国タンデムに転属. 1991年12月インフォストラテジ設立し代表取締役. 「フォールト・トレラント・システム」編訳 (1986年). 「OLTP システム」編訳著 (1991年) ほか. 電子情報通信学会会員. IEEE 会員.