ＡＤＡＢＡＳの分散データベース機能

石井義興　嶋田正裕
㈱ソフトウェア・エージー

1．はじめに

ＡＤＡＢＡＳに付加さる分散データベース機能について述べる。ＡＤＡＢＡＳの分散ＤＢ機能の特徴は、すでに分散処理されているものを統合するというよりはむしろ、現在集中処理されているものを分割／分散化し、メリットを得ようという所にある。この際、ユーザ・アプリケーション・プログラムはこの分割／分散化に際してまったく変更する必要はない。すなわちデータの分布が変化しても利用者が意識しなくてもすむということ、プログラムの組み直しをしないですむということである。このようなフレキシビリティをＤＢＭＳに持たせることは必ずしも容易ではないが、今後の発展を考えれば必須の機能である。

分割／分散化しているマシン（ノード）を接続する手段は単に通信回線によるのみではなく、チャネル結合でも良く、また仮想マシン間でも良い。各ノードを接続するためのソフトウェアも開発済みである。各ノードとなり得るマシンはＩＢＭ３７０、３０８Ｘ、４３ＸＸ、日立、富士通のＭシリーズで、ＯＳは、ＭＶＳ、ＭＶＳ／ＸＡ、ＶＳ１、ＤＯＳ／ＶＳＥ、ＶＯＳ１、ＶＯＳ１／ＥＳ、ＶＯＳ２、ＶＯＳ３、およびＯＳⅣ／Ｆ４である。異なるメーカのハードを利用し、異なるＯＳを用いても良い。ＡＤＡＢＡＳの分散データベース機能、ノード間接続ソフトウェアの総称はＮＥＴ－ＷＯＲＫという名で呼ばれており、ＮＥＴ－ＷＯＲＫは多くのコンポーネントから構成されており、各コンポーネントには、それぞれ名前が付いている。

2．集中処理の問題点

コンピュータの利用はバッチ処理から始まった。この事は、コンピュータの歴史を振返ってみればすぐわかる。昔はコンピュータの価格が高く、したがって多くのコンピュータを設置することなど不可能であった。そのためデータを集め、バッチ化して処理することとなった。集中処理が当然のように思われ、しかも、それが最良の方法であるかのごとく考えられていた時代さえあった。しかし企業体の中での活動は、そもそも人手によって分散処理されており、全社の処理が１ヵ所で集中的に処理されているわけではない。

人手によって分散処理されている作業をコンピュータライズするわけであるから、分散処理していた時の問題点を集中化によってカバーする事が当然要求され、システム・デザイン時にそれを加味するように考えられた。しかしその際、分散処理の利点も同時に失われるケースが多かった。現在は集中処理による問題点をむしろ分析し、分散処理されるべきものは、分散処理することによって、集中と分散のバランスを取り、最良のシステムとなるように考え直す時期にきている。

集中処理による問題点は次の通り。

◇バッチ処理による集中化はタイムラグが大きくなる。すなわち、ある部分では準備完了の状態となっても、どこかの部分のデータ準備がととのわないと、それによって処理が待たされる。

◇バッチ処理による集中化では、現場でのデータ・ハンドリング作業と中央での作業と２度手間になる。

この問題を解決するためには、バッチ処理をやめ、オンライン方式に切り換えれば一応かたずく。しかしオンライン方式による集中化は一般に非常にコンピュータ資源をくいコスト高となる。

現時点は、このオンラインによる集中処理方式から、一部分散処理へもどすことによって、コストを削減し、スピード・アップをはかろうと考え始めた段階といえる。

3．中型複数コンピュータによるシステムの有効性

「コンピュータのコスト・パフォーマンスは大型機になればなる程良い」という神話はＩＢＭ４３００シリーズの出現と共に崩壊し

た。ＩＢＭを初めとする各コンピュータ・メーカは常に新しいコンピュータ・シリーズを世に出し、しかも大型機を利用すればする程コスト・パフォーマンスは良かった。しかし表１に示す通り、４３００シリーズ（中型機）のコスト・パフォーマンスはＩＢＭの超大型コンピュータより安い。このことは中型コンピュータを複数台組合せることによって大型コンピュータの代りに用いることがコスト的にも見合うことを示している。しかもマシン・ダウン対策を含めて考えると、複数台の組合せによる方式の方がはるかにリライアブルである。

コストがかえって安くなる点は「大量生産によるコンピュータ（小型機）の方が大量生産でないコンピュータ（超大型機）より安い」ということから考えれば当然のことと見ることができ、今後もこの傾向は続くと考えられる。これによりコンピュータを複数台連結し、大型コンピュータの代りに使用するという考え方が成立する。

コンピュータの処理能力がもし限界に達したら今までのように単にマシンをグレード・アップするのではなく、「小型のＣＰＵをＨＯＳＴマシンに付加することにより処理量の増大に対応しようという考え方もなりたつのである。

このことは分割／分散処理がコスト面で優位に立つことを示している。特にオンライン・リアルタイム・システムを構成する場合はバックアップ・マシンを必要とする。バックアップ・マシンの能力はメイン・マシンとほぼ同程度の能力を持つ必要がある。メイン・マシン１台のコストで中規模マシンを複数台買入すれば、バックアップ・マシンを買入する必要がない。（図１参照）なぜなら３台のマシンが同時にマシンダウンを起すことは考えられないからである。

ただしこのシステムが正しく稼動するためには、３ヵ所に同一ユーザ・プログラムを入れることになるし、１つのＤＢを３台のマシンで同時にＵＰＤＡＴＥできなければならない。つまりＤＢＭＳ側にこれを可能にする機能が必要となる。

	4331-2 1M	4341-2 4M	3033N 8M	3081 16M
ＭＩＰＳ	4.0	1.2	3.7	10.4
価格（千ドル）	150	416	1.695	3.720
３０８１　１台の値段で何台買えるか	25台	9台	2台	1台
合計メモリ・サイズ	25M	36M	16M	16M
合計ＭＩＰＳ	10	10.8	7.4	10.4

表１

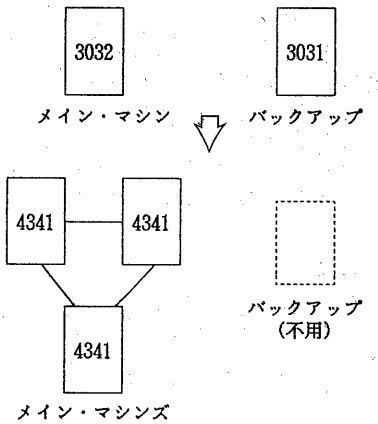


図１

４．分散処理、分散ＤＢの実際例

今後の分散処理環境を予想し、㈱ソフトウェア・エージーでは図２で示すようなマシン構成を作り、ＮＥＴ－ＷＯＲＫの開発テストを行なっている。わざわざ異なるメーカのマシンを導入し、異なるＯＳを入れている。現時点ではＶＡＸ－１１とＩＢＭはまだ結合されていない。

この構成の特徴はデータベースのあるノードとそれをアクセスするプログラムがランするノードが異なってもかまわず、それぞれがユーザ・ニーズに対応して移動しても良い事を前提に作られたものである。

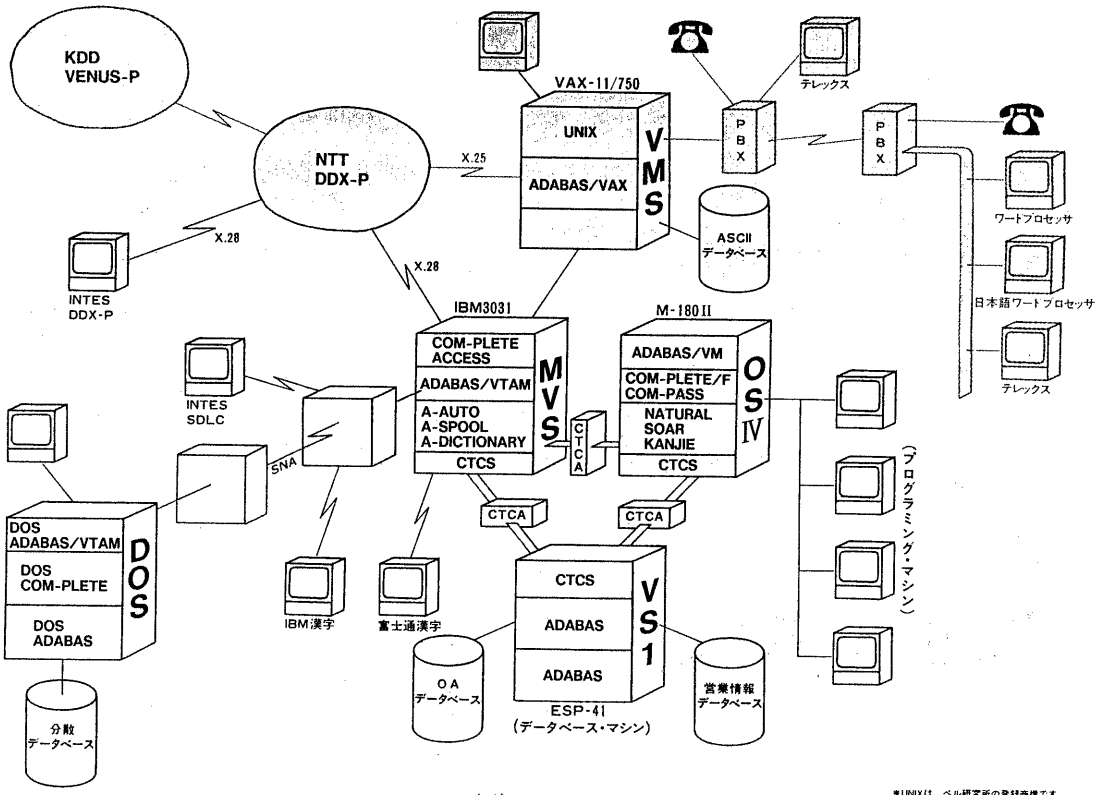


図2

図3に示す例は野村證券のＣＡＰＩＴＡＬである。このシステムは集中処理をする際処理量が多く、メーカが提供する最大機器構成を用いても処理不能なため、処理を分割し、3台のＣＰＵを結合し、1つのＤＢをアクセス／更新する分散処理システムを構成した例である。ＣＡＰＩＴＡＬの場合は分散ＤＢとはなっていないが、ＤＢに対するアクセス／更新がさらに多量になった場合はＤＢを分割し、分散ＤＢ化し、それぞれのＤＢに１ＣＰＵを対応させることによって処理負荷を軽減させ、システムのニーズに対応させ成長されることができる。

5．分散処理における問題点

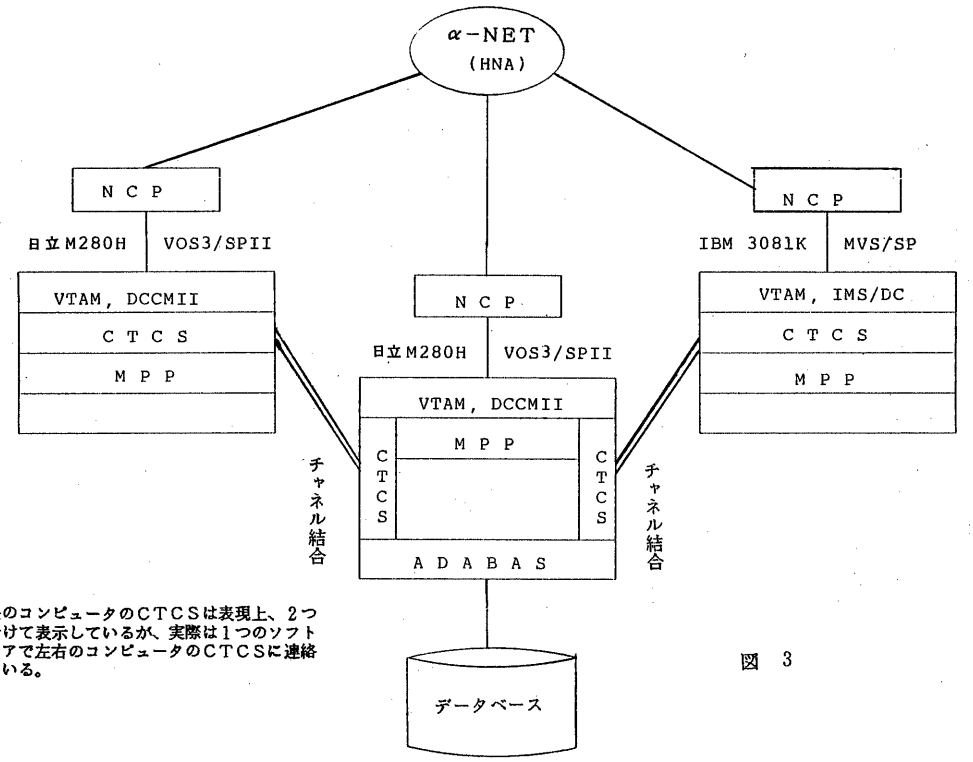
1）一般的な考慮点

◇分散処理を構成する各コンポーネント、たとえば、ＯＳ、データベース管理システム、コミュニケーション・ソフトウェア、エンドユーザ言語等は非同期に各々バージョン・アップされ、しかもそれらは統一的考え方にそって変更されるとはかぎらない。これにどう対処するか。

◇分散処理は全体が完成することはなく、ステップ・バイ・ステップに変化しながら稼動しなければならない。このような柔軟性をどう保証するか。

◇分散処理環境を作り上げてゆくプロセスで、単独に稼動しているシステムに悪影響を与えないようにするにはどうするか。

◇将来提供される新らしいコミュニケーション方式や、ディスクにも最少の労力で対応できることが望ましい。



●中央のコンピュータのＣＴＣＳは表現上、２つに分けて表示しているが、実際は１つのソフトウェアで左右のコンピュータのＣＴＣＳに連絡している。

図　3

●ＣＡＰＩＴＡＬサービスで構築したマルチホスト単一ＤＢ方式の概念図。中央のＭ－２８０Ｈに證券情報を提供するためのＡＤＡＢＡＳデータベースがあり、２台のホスト・コンピュータ（Ｍ－２８０ＨとＩ－３０８１Ｋ）がチャネル結合する。図の四角で囲んだのは主要なソフトウェアで、ＮＣＰ（Network Communication Program）はネットワーク通信プログラム、ＶＴＡＭとＤＣＣＭ－ＩＩは通信プログラム、ＭＰＰはメッセージ処理プログラム、ＣＴＣＳにチャネル結合プログラムである。

2）分散処理環境を実現するための機能

◇分散ネットワークのダイナミックな再構成たとえば、ノードの位置、アプリケーションの位置の再構成があっても簡単に対応できること。

◇孤立ファイルを取扱う機能のみならず、重複ファイル、分散ファイルを取扱うことも可能なこと。孤立、重複、分散ファイルの定義は次でふれる。

◇データベースの管理、ネットワークの管理は、どこのノードからでも可能なこと。

◇各ノードのすべてにシステム担当者や、アプリケーション担当者を置かなくても良いこと。

◇メッセージ・ジャーナリング、ファイルあるいはデータベース・リカバリ、パフォーマンス・チューニング、アカウンティングおよびセキュリティの定義／変更がローカルにもグローバルにも行なうことができること。

◇遠隔地のコンピュータ資源をローカルにも使用できること。

◇コンピュータ間のパスやコンピュータ自身がダウンしても全メッセージの送信を保証し、全トランザクションを完了させられること。

◇ネットワーク内の代理パスやバックアップ・パスをダイナミックに選択できること。

◇仮想マシン・ノードを置き、それによってテストが可能なこと。

◇非分散化アプリケーションに対しては上位方向の互換性があること。

6．分散データ環境における ファイルのタイプ

分散処理環境内のデータ・ファイルには次の３つのファイルタイプを考える必要がある。

◇孤立ファイル（普通のファイル）
分散処理環境内に１個だけ存在し、それは１個のデータベース中にのみある。普通のファイルはこのタイプである。

◇重複ファイル
分散処理環境内に同一のファイルが重複して２個以上存在し、それぞれ別のデータベース内にある。

◇分散ファイル
同一タイプのレコードが物理的に複数のファイルに分割されており、ユーザは論理的には、１個のファイルと認識しているもの。

孤立ファイルはいわゆる普通のファイルのことで、重複ファイルは分散処理の際ひんぱんに使用されるマスタ・ファイル類のことであり、重複してデータをもつことによって遠くまでデータをリファーしに行くことなくアクセスのスピード・アップをはかろうというタイプのファイルである。分散ファイルとはいわゆる伝票類を入れるファイルのことで、伝票は各部所に分散して発生するが、このタイプのデータは分散ファイルに入れることになる。

7．論理ＤＢ、ファイルと 物理ＤＢ、ファイル

ユーザあるいはユーザ・プログラムが認知するＤＢ、ファイルを論理ＤＢ、ファイルと名付ける。１論理ＤＢ、ファイルは複数物理ＤＢ、ファイルに分割され格納されても良い。物理ＤＢとは１ＡＤＡＢＡＳ　ＤＢＭＳによって管理されるＤＢを指し、その物理ＤＢ内の各ファイルを物理ファイルと呼ぶ。集中処理されている状態は論理ＤＢと物理ＤＢがまったく同一と考える。集中処理されているＤＢ（論理）を分割（物理的）し、分散ＤＢとしても、論理的には１ＤＢとみる。論理的には今までと同一であるから、そのＤＢをアクセス／更新するユーザ・アプリケーションは変更する必要はない。集中処理を分割し分散ＤＢ化した簡単な例を図４に示す。

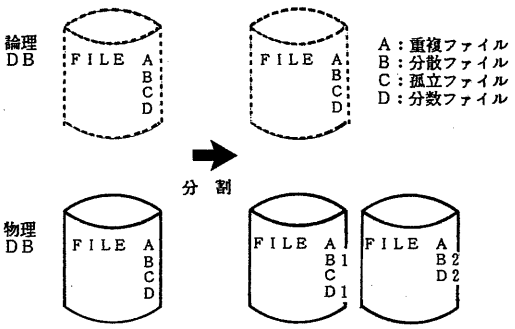


図　4

論理ＤＢ、ファイルと物理ＤＢファイルとの対応はＡＤＡＢＡＳの中のＡＤＡＮＥＴと呼ばれるコンポーネントによってダイナミックに行なわれる（図５参照）。物理ＤＢは同一ノード中にあって良いし、異なるノード中にあっても良く、また仮想マシン上にあっても良い。異なるノード中にある場合、両者はチャネル結合（ＣＴＣＡ）あるいは通信回線（ＶＴＡＭ）で結合されていなければならない。仮想マシンの時はＶＭを利用し、ＶＭ下にはＣＭＳ、ＤＯＳ／ＶＳＥ、ＶＳ１、ＭＶＳ等多種類のＯＳ下にそれぞれＡＤＡＢＡＳを入れ、それらの間で自由にコミュニケートできる。これらの機能を実現するためにＣＴＣＳ、ＡＤＡＢＡＳ／ＶＴＡＭ、ＡＤＡＢＡＳ／ＶＭと名付けられたソフトウェア・プロダクトが開発されすでに利用されている。

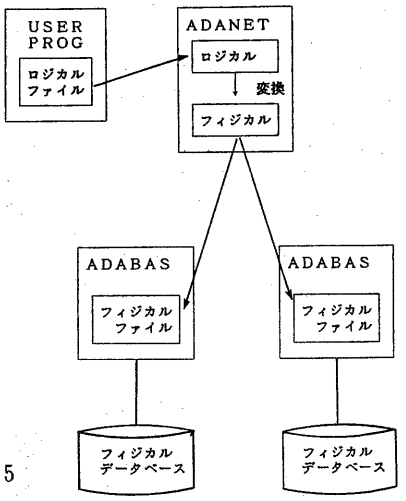


図　5

8．処理の分割／分散化と
データベースの分割／分散化のプロセス

ＡＤＡＢＡＳを利用し集中処理されているデータ処理を分割／分散化するプロセスをここで示す。図６はチャネル結合（ＣＴＣＡ）を利用し処理を分割した例である。ＳＴＥＰ１は集中処理の段階である。ＳＴＥＰ２はＳＴＥＰ１でＣＰＵ不足になった際、ＤＢＭＳのＣＰＵ負荷をバックエンドＣＰＵに移動させ、ＣＰＵ負荷を軽減させる例であり、ＡＤＡＢＡＳデータベース・マシンはこの考えを実現したものである。マシン間の結合のためのソフトウェアはＣＴＣＳと名付けられている。ＳＴＥＰ３は処理をさらに２分割した例である。

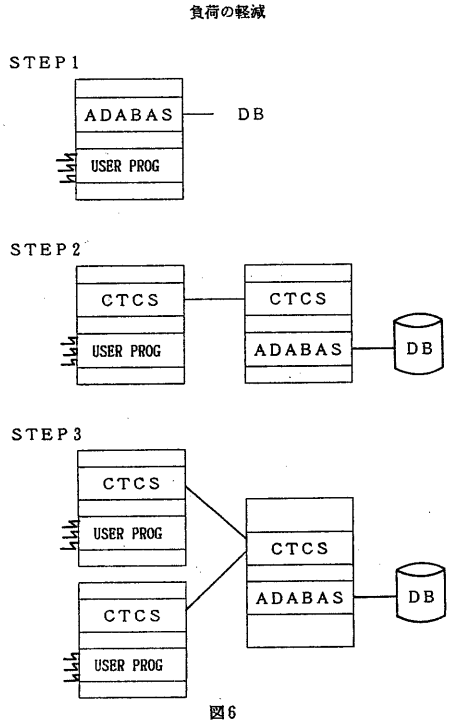


図6

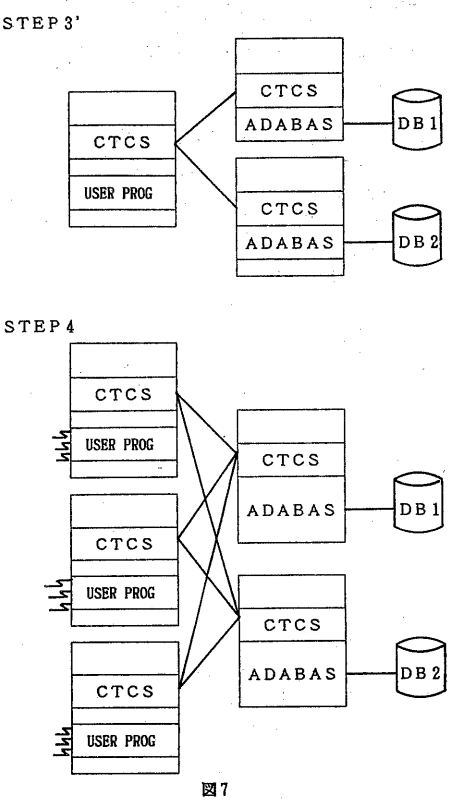


図7

図３に示した野村證券のＣＡＰＩＴＡＬはこの例である。ＳＴＥＰ２、３はデータベースが分割／分散化されてはいない。図７のＳＴＥＰ３’、ＳＴＥＰ４は処理／ＤＢＭＳ負荷がさらに増大した時にデータベースを分割／分散化した例である。いずれの場合もユーザ・アプリケーションを変更する必要はない。

図８は通信回線を利用し遠隔地からもデータベースへのアクセス／更新を行なう例である。ＳＴＥＰ１は集中処理である。ＳＴＥＰ２は遠隔地からもデータベースをアクセス／更新する例であり、マシン間のコミュニケーションはＶＴＡＭを利用し、ＡＤＡＢＡＳ／ＶＴＡＭと名付けられたソフトウェアがユーザ・プログラムから出されたＡＤＡＢＡＳに対するリクエストをデータベースのあるマシンへ転送する。ＳＴＥＰ３はデータベースが分散化されている場合の例である。この場合もユーザ・アプリケーションはデータベースがある場所を意識する必要がない。

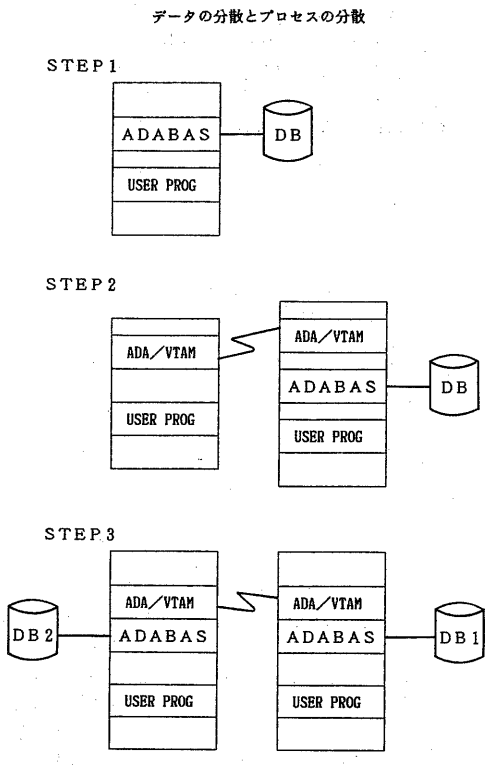


図8

９．分散環境内の問い合せ処理

孤立ファイル

孤立ファイルについては、ノードおよびレコードを選択したいファイルをＤＤ（データディクショナリ）を利用しユニークに識別できる。したがって高レベル・エンド・ユーザ言語では論理ファイル名を用いてコーディングすればよい。したがって論理ファイルに対応するすべてのファイルが孤立ファイルの時は、必要に応じてリクエストを転送し、各ノードやＤＢＭＳから返されるレコードをユーザ・プログラムへリターンすれば良い。

分散ファイル

分散ファイルはいくつかのノード「論理的には１個のファイル」が分散しているものをいうが、ＤＤには単に、あるファイルが分散ファイルであるという情報を持たせるのみならず、いかなるロジックでデータが分散しているかのロジックも同時に入っている。一番簡単なロジックは各物理ファイル内でユニークにつけられる番号（一般にタブル・ＩＤと呼ばれる分散ファイル全体でユニークな番号など）によって分けるロジックである。ユーザ・プログラムから分散ファイルに対して検索する場合はＤＤからどれとどれが各々の物理的分散ファイルであるかを判断し、各々の物理ファイルに対し検索する。ＡＤＡＮＥＴは答を単にマージすれば良く、マージされたものをユーザ・プログラムへ返せば良い。各レコードのリードはＡＤＡＮＥＴによって判断され必要に応じて他のノードへリクエストが転送され、対応レコードが読まれる。

重複ファイル

重複ファイルにたいする検索はユーザ・プログラムが稼動しているコンピュータ（オリジネーティッグ・ノード）にもしその重複ファイルがあれば、単にそれに検索すればよい。もしオリジネーティング・ノードに重複ファイルがないときは一番近い重複ファイルへリクエストが転送されアクセスされる。

10. 統合化された分散データベースの更新

データベースが物理的に分散した位置にあり、しかもそのいくつかの物理データベースにまたがったデータ更新を１論理トランザクションで行なう場合は一般にマシン・ダウン、ＤＢＭＳアベンド、回線ダウン、ＴＰモニタ・アベンド等の問題が起るとデータ間の不整合を起す可能性がでてくる。

この問題を完全に解決するためには、まず各ノードのすべてのＤＢＭＳが、まず完全なリカバリ機能を持っている必要がある。その上でさらにマルチ・ノード間でのダウンによる不整合をオートマチックに修復する機能をＡＤＡＢＡＳ、ＡＤＡＮＥＴに持たせなければならない。マルチ・ノード間でのリカバリに関しては11．で述べる。

以下まず排他制御に関して述べよう。排他制御を実施するための機能としては普通行なわれているユーザ占有機能を用いる。

ユーザ占有（レコードのホールド）の単位は、すなわち１トランザクション内でアップデートのためにロック（ホールド）する最小単位はデータ・レコードとする。こうすれば容易に分散データベース環境にもユーザ占有概念を拡張できる。ホールド要求については更新のためのレコードを読む時にのみ必要であり、ユーザ占有の設定にＤＢＭＳ内で消費されるオーバヘッドはたいしたものでない。

ホールド付の読み込み命令が、あるノードに対して発行される場合、個々のレコードのホールド状態はそのローカル・ノード内で設定すればよい。すなわち、個々のユーザに対して、ホールドしたレコードに関する複数のタプルＩＤリストがいくつかの別々のノードのＤＢＭＳ内に存在することになる。トランザクション処理中に、もし、ユーザ・プログラムとあるノードとの間のコミュニケーション問題が生じた場合は、そのユーザがホールドした全レコードを、ＡＤＡＢＡＳが自動的に開放すればよい。

ファイルのタイプごとに、ユーザ占有を設定する方法は、次の通り。

◇孤立ファイルについては、データ・レコードは常に１つのユニークに識別されるノードから選択されるので、そのレコードをホールドする。

◇重複ファイルについては、重複ファイルがあることをユーザは全く意識する必要はない。重複ファイルのあるレコードに対してホールド要求があると、全重複ファイルのレコードを自動的にホールドする。ユーザは重複ファイルの全てについて該当レコードがホールド状態になったとき、リクエストに対する答を受け取るだけである。この場合ユーザ占有は１個別レコードについて複数ノードにわたることになる。重複ファイル内でユーザ・レコードがホールドできないと、ユーザはレコードがホールド可能となるまで待たされる。もしデッドロック状態になるとトランザクションを排他せよという要求をＡＤＡＮＥＴがユーザ・プログラムへ返せばよい。

そのトランザクション処理に関係するすべてのレコードがホールドされるとユーザ・プログラムではホールドされた各レコードに対し更新処理を行なうことができる。更新処理に関しても各ファイル・タイプにそってそれぞれは次の状態を扱わなければならない。

◇孤立ファイルおよび分散ファイルについては、分散データベース内で単一のレコードだけが更新される。

◇重複ファイルは２つの方法で更新される。重複ファイルはＤＤ内に即更新かオフライン更新と定義できる。即更新とは、重複ファイルが存在するいずれかのノード内でファイルが更新されると、これら重複ファイルも同一トランザクション内で直ちに更新されることを示す。オフラインとは、オンラインで更新すべき分散データベース内のファイルは１マスタ・ファイルのみであり、ある特定時点でマスタ・ファイルをコピーするまで重複ファイルはそのままであることを示す。即更新については、ユーザ占有は前記の全重複ファイルに対して設定される。実際の更新操作では、ユーザが発行する更新コマンドは重複ファイルの全コピーに対し

てＡＤＡＮＥＴからダブって発行される。更新動作に入る前に重複ファイルの全コピーに対してユーザ占有が設定されるので、全ファイルは同期がとれユーザ干渉（インタフェランス）は起こらないことが自動的に保証される。このような方法をとれば、ユーザにとってトランザクション処理は単一のデータベース環境の場合と同じである。

11. 分散環境における トランザクション処理と同期化

非分散環境と同様、自動的に論理トランザクション・レベルで自動的リカバリ・リスタートを扱う方法を考えよう。

ユーザ・プログラムがＤＢＭＳと同一のマシン上で稼動する単一データベース・システムとは異なり、分散データベース環境では複数コンポーネントを含む。

◇各データ・ノード内のプロセッサと周辺機器

◇データ・ノード間の通信回線

単一データ、単一プロセッサ環境との主な相違は、コンポーネントの１つに障害があった場合、ネットワーク全体が操作不能になるわけではないことである。個々のノードは操作を続行できる。

ネットワーク内で、ユーザ・プログラムやリクエストを処理するノードをオリジネーティング・ノードといい、トランザクションに含まれるリモート・データベースを持つ全ノードはターゲット・ノードという。

オリジネーティング・ノード以外のターゲット・ノードのマシン・ダウン、オリジネーティング・ノードとターゲット・ノードとの間の回線ダウンなどいろいろのケースについても、トランザクション・リカバリ機能を持たねばならない。これは単に各ノード内のＤＢＭＳが単独でリカバリ機能を有しているのみでは問題を解決することができない。

この問題を解決するために次のアーキテクチュアを提案する。

まず各ＤＢＭＳは END OF TRANSACTION COMMAND をうけるようにする。ＤＢＭＳによっては内部的にＧＥＴによってターミナルからデータを受け取った時それ以前のトランザクション処理を「終了」とみなすものがあるが、ここで提案する方法では END OF TRANSACTION コマンドを出してもらうようにする。

その上でＡＤＡＢＡＳ、ＡＤＡＮＥＴはその END OF TRANSACTION （以下ＥＴと略す）を受取った時、次の手順で処理する。

①ユーザ・プログラムはＥＴコマンドを発行する。

②ＰＥＴ（フレリミナリＥＴ）コマンドをＡＤＡＮＥＴが自動的に生成し、そのトランザクションに関連する全ノードに対し発行する。オリジネーティング・ノード内のＡＤＡＮＥＴは、どのターゲット・ノードに、ＰＥＴコマンドを発行したかの全ノードのリストをディスクにセーブする。ＰＥＴコマンドは、戻し処理できるようログ・データに対する全Ｉ／０を生成する。ＰＥＴコマンドはユーザがホールドしているレコードは開放しない。

③ターゲット・ノードに対する全ＰＥＴコマンドにＯＫの応答があると、ＥＴコマンドがＡＤＡＮＥＴによって並行して全ノードに対して送られる。当ＥＴコマンドは全ノード内で、ユーザがホールドしていた全リソースを開放する。もしターゲット・ノードの１つから応答がないと、他の全ノードに対してトランザクションのバックアウト（戻し処理）指示がＡＤＡＮＥＴから出され、オリジネーティング・ノードのユーザはバックアウトしたことを知らされる。

④全ノードから応答を受取ると、ユーザにトランザクション終了の旨知らされる。

当手法では、最小のオーバヘッドで分散処理環境にトランザクション処理を設定できる。

このトランザクション処理に含まれるノードやパスに起こり得る障害や、障害後のリスタートは、１つのノードでトランザクション処理を行なうと同様を扱える。ネットワーク内の障害の最悪のケースは、ＡＤＡＮＥＴが自動的に出すＰＥＴと最終のＥＴコマンド間の時間である。

トランザクションに含まれる全ノードに並行してＰＥＴおよびＥＴコマンドが送られたとき、この時間は最小となる。これは５００マイクロ秒台であろう。なぜならＰＥＴではすべて I／0 がすでに出されているからこの間の時間は多くはない。しかしこの間に回線が操作不能となるときは、ターゲット・ノードはトランザクションを終了すべきかバックアウトすべきか決められない。そこでＡＤＡＮＥＴがまずオリジネーティング・ノードとコミュニケーションしトランザクションの状態を確認する。

すなわちトランザクションの待ち時間中に回線が操作可能となるかもしれないし、この場合にはターゲット・ノードは停止中のトランザクションの終了方法を決めるのに必要な情報をＡＤＡＮＥＴから自動的に受け取る。オリジネーティング・ノードとコミュニケーションが確立できないと、停止中のトランザクションがオペレータ・コンソールに表示される。

次の情報を表示する。

◇トランザクション番号

◇オリジネーティング・ノード

◇他のターゲット・ノード

この場合、オペレータは再動作するのに次のどれかの方法をとればよい。

◇回線が間もなく操作可能になると予想できればトランザクションを待ち状態にする。

◇電話でオリジネーティング・ノードのオペレータを呼び出し、特定トランザクションのステータスをきく。

◇トランザクションをバックアウトするか終了するかを任意に決定する。この場合、当トランザクションの全更新操作を特定のログ上にロギングしておけば、後で必要なときこの全情報を使用すればよい。あるノード内のプロセッサのリスタートについては、停止中のトランザクションのバックアウト／終了処理に対する全コミュニケーションも自動的に行なわれ、リスタートは単一プロセッサ環境のように扱える。

12. おわりに

ＡＤＡＢＡＳ分散処理／分散データベース機能について述べた。ＡＤＡＢＡＳの分散データベース機能は集中処理から処理と、データベースの両方を分割／分散化することによって全体的処理効率を向上させ、かつ、フレキシビリティを増大させようというねらいを持っている。日本においてもすでにこの方法は実用化され稼動している。今後大量生産に伴う中規模マシンのコストが大巾に安くなることが予想され、処理とデータベースの分割／分散化の方向は益々確乎たるものとなるであろう。