データベース・システム 77－1
（1990．5．17）

同報指向分散データベースシステムの研究開発

小寺誠、坂本明史、川上英、疋田定幸
（沖電気工業株式会社）

同報通信をベースにした分散データベースシステムの研究開発に関して報告する。集中型データベースに比較して、分散データベースは通信コストの面で不利である。著者らのアプローチは、同報通信の特性の利用により通信コストを削減し、分散データベースを実現しようとするものである。本稿では、同報通信を利用した検索処理、および更新処理の実現を概観する。

Research and Development of a Broadcast Based Distributed Database System

Makoto KOTERA, Akifumi SAKAMOTO, Suguru KAWAKAMI
and
Sadayuki HIKITA

Oki Electric Industry Co., Ltd.

16-8, Chuo 1-chome, Warabi-shi, Saitama 335, Japan

The authors report research and development of a broadcast based distributed database system. In comparison with centralized databases, distributed databases are at a disadvantage in both implementation and performance mainly due to communication overhead. The Authors' approach is to implement a distributed database based on a broadcast network. By taking advantage of broadcasting, it is expected to reduce communication overhead enough to make a distributed database realistic. In this paper, the authors would sketch out the outline of the result from their three-year research and development work on this approach. This paper presents the broadcast oriented database retreival mechanism in the first half. Then, the authors mention the problems in updating distributed data consistently, by focusing on global deadlock problem.

１．　はじめに

分散データベースシステムは、複数のプロセサ（以後、サイト）が協調動作することによって、すべてのサイトにあるデータをあたかも一つのサイトにあるかのように利用できるデータベースシステムである。分散データベースシステムは、地理的に分散したサイト間でのデータの共有、負荷の分散、情報を集中することに対する危険性の回避などの利点があり、近年の計算機ネットワークの発達にともなっていくつかの実験システムの実現が試みられてきた。しかし、まだ技術的課題が多く残されており、現在までに実用化されたシステムはない。

分散データベースシステムの主要な利点は、以下に示す３項目である。第一は、負荷の分散である。従来、単一のプロセサで実行していたデータベースの処理を複数のプロセサに分散することで、過度の負荷集中を避けることができる。第二の利点は、負荷分散に関連している。すなわち、複数プロセサによる並列処理である。特に、ネットワークで接続された疎結合型のマルチプロセサアーキテクチャ上での分散データベースシステムは、密結合型のマルチプロセサアーキテクチャ上のデータベースシステムに対して、処理上のボトルネックが生じにくい点でも有利である。ただし、前提としてプロセサ間を接続するネットワークの速度が十分に高速である必要がある。第三の点は、拡張性である。ネットワークへのプロセサの接続は、バスへの接続に比較してはるかに容易である。これは、最初は少数のプロセサで構成されていた分散データベースシステムを、処理の拡大にしたがって逐次に拡張できることを示している。

分散データベースの利点を十分に生かすためには、解決されなければならない問題が多く残されている。その中で最大の問題は、分散データベースを構成するプロセサ（以下、サイト）を協調動作させるための通信コストである。近年のネットワーク技術の発達も、これらの通信コストを、無視できるほど十分には小さくしていない。通信コストにかかわる問題の解決のためには、なお多くの努力を払わなければならない。

本稿では、同報指向分散データベースの研究開発において、これまで著者らが得た成果を報告する。２章で分散データベースにおける問題点を概観した後、同報ネットワークをベースにした分散データベースのモデルを記述する。３章では、検索処理に関して、これまでに得ることのできた成果を、同報をベースにした通信方式と問い合わせ処理に焦点をあてて述べる。そのあと、４章において分散データベースにおける、もう１つの重要な処理、分散更新処理に関して、グローバルデッドロック問題を中心に課題とアプローチ、および成果を論じるものとする。

２．　分散データベースシステム

２．１　課題

分散データベースシステムの実現上、最大の困難を招くのが、複数サイトを結合するための通信コストである。具体的には、検索演算やシステムの各種管理情報へのアクセスに必要な通信のコストを意味する。

通信コストには２つの面が存在する。１つは、ネットワーク上のデータ転送に伴う通信時間であり、ネットワーク媒体の速度に依存する。もう１つは、データ通信に伴う固定的に必要な通信処理（プロトコル・ハンドリングなど）時間である。ネットワークの高速化が進むにつれて、後者のコスト削減がより重要となってくる。データ通信回線が低速で、信頼性の低かった時代、研究の主眼は、転送データ量の減少による通信時間の削減に置かれていた［HBVN79］。しかし、ネットワークの高速化は、準備時間の通信コストに占める比率をより高いものとしつつある。事実、分散データベースの処理効率が、通信準備に要する処理時間に対して極端に敏感であるとの研究結果も報告されている［BHID88］。これは、通信コストの研究は、通信量の削減よりも、通信回数の削減に焦点を当てるべきであることを意味している。

集中データベースシステムに対する上記のような有利さにもかかわらず、通信処理が必須となる分散データベースシステムの実現は決して容易ではない。数年以内には、超高速で、高信頼性の光ネットワークが実用化される可能性がある。しかし、従来、そして現在、分散データベースシステム実現のために利用可能なネットワークのデータ転送速度は、バスの速度に比べればはるかに低速である。依然として、通信は単一の計算機内での処理に比べて、より時間のかかる処理である。通信処理に必要な時間（通信コスト）を、より削減する方式の開発が、分散データベース実用化のための主要な課題となるのである。

２．２　データベースのモデル

ここでは、３章と４章の報告のベースとなっている分散データベースシステムのモデルを記述する。モデルとなる分散データベースは、同報通信路で接続されたｎ個のサイトの集合｛S_1……S_n｝から構成される。ｍ個の親トランザクションが存在し、t_1 ……t_m と表記される。サイトS_i上のトランザクションt_iが、他サイトS_k上のデータベースをアクセスする必要が生じたとき、そのサイト上に子トランザクションt_{ik}を発生させる。各サイトのローカルなデータベース管理システム（ＬＤＢＭＳ）は、サイト上で閉じたデッドロック（ローカルデッドロック）を検出する機能を持つと仮定する。

3．　分散データベースシステムへの同報指向アプローチ

著者らの基本的なアプローチは、同報通信を利用した通信コストの低減である。同報性を持つネットワークの代表例として衛星通信回線やＬＡＮがある。このようなネットワークでは、あるサイトから送出されたデータは、他のすべてのサイトに届くという性質を持つ。この性質を利用すれば、より少ない通信回数で分散データベースの制御を行う方式を開発できる可能性がある。

同報通信路の利点が期待される一方、同報通信路の持つ不都合な特性に対する研究はまだ不十分である。分散データベースの効率的な制御に不可欠な、上位のプロトコルに関する研究はほとんど行われてこなかった。また、同報通信をベースとした分散データベース構築のためには、問い合わせ処理、分散更新制御の実現などの観点からの検討も必須である。同報通信路へのデータ送出の衝突の回避に関しては、従来多くの研究が行われている。コミットメント制御、および重複データの制御の効率化に、同報通信の利点を生かそうとする研究は、すでに行なわれている［CHAN83］。しかし、従来の技術を単純に同報通信路に適用するだけで通信コストの問題が解決できるわけではない。

同報通信路の問題点と、それぞれに対する著者らのアプローチを示す。問題点の第１は、同報通信でのリライアブルなデータ転送の方式が解決されていない点にある。これは、通信アーキテクチャにかかわる問題である。第２の問題は、通常同報通信が同一メッセージを複数サイトのデータベース制御部に送る場合だけに利用される点であり、分散データベースにおける問い合わせ処理にかかわってくる。さらに第３の問題は、同報通信の特質が引き起こす副作用の問題である。ここで副作用とは、全サイトに無条件に通信パケットが送達されるために、各サイトにかかる負荷がきわめて大きなものとなる問題を意味する。

ここでは、本研究における同報指向アプローチのもたらした成果を、３つの観点から述べる。すなわち、通信アーキテクチャ、問い合わせ処理、およびマルチキャスト指向の通信プロトコルである。

3．1　コネクションベースの同報通信

リライアブルな転送処理には、通信相手との論理的なコネクションが必須である。コネクションを設定することにより通信の端点は相手端点に関する情報を保持でき、これに基づいてデータの紛失の検出、および再送を行うことができる。

同報通信を用いた分散データベース制御のためには、リライアブルな同報通信を実現する制御機構を開発しなければならない。同報通信は、コネクションレスのサービスを原則としている。メッセージが紛失した場合の再送などは行われていない。再送のための制御を、単純にコネクションレスの同報通信をベースにして実現するものとすれば、不必要なブロードキャストパケットがますます増加することになる。つまり、再送の必要のないサイトにまで再送要求が送られてしまうのである。再送要求を受け取ったサイトは、本来、不必要であるにもかかわらず、新たなブロードキャストパケットを送信し、ネットワーク上のトラフィックとネットワークに接続されたサイトの負荷を増大させる結果となる。通信媒体が高い信頼性を保証していたとしても、ネットワーク上を流れるブロードキャストパケット数の増加は、パケットの紛失と、それにともなう再送の発生の確率を高める。このため、信頼性の高いサービスを実現できず、メッセージの到着順序に意味のある場合や、データベースのタプルなど、メッセージ紛失が起きると困る性質のデータを送信することができない。

現在のところ、同報通信でのコネクション設定という考え方は、一般的なものとは考えられていない。そこで、本研究では、通信相手が複数であるコネクションの概念を導入し、これを同報コネクションとよぶことにする。さらに、分散データベース制御のために必要な同報コネクションのプリミティブを検討することにより、マルチキャスティングによりリライアブルな通信を実現する制御方式を開発した［坂斎88］。

同報コネクションの設定・解放のプリミティブを、従来の１：１通信のものを拡張して考えることができる。通信相手が一人である１：１通信では、コネクションの設定・解放は単純である。それぞれ「相手を指定してのコネクション設定」、「コネクション解放」のプリミティブによって実現できる。これらのプリミティブを同報通信用に拡張すると「複数の相手を指定しての同報コネクション設定」、「同報コネクション解放」となる。既存の実験的なシステムでは、この機能によって同報通信を実現しているものがある。

この方式の欠点は、通信相手の動的な変化への柔軟さに欠ける点である。アドホックな問い合わせ処理では、通信相手のサイトがあらかじめ特定できない。サイトS_2、S_3への同報コネクションを設定して処理を行っている途中で、あらたにサイトS_4へのコネクションが必要となる場合を考える。上記のプリミティブでは、いったん同報コネクションを解放した後再度コネクションを設定しなければならない。そうでなければ、新たに生成するサイトS_4だけからなる同報コネクションは、サイトS_2、S_3の同報コネクションとは別のものとなり、サイトS_2、S_3、S_4への同報通信を実現することができない（図１(a)）。

したがって、本研究では、同報通信のプリミティブのために、必要に応じてコネクションを動的に変化できる機構を開発した。これは、同報コネクションへサイトＳ　のようなメンバの追加と削除を許す（図１(b)）ものである。次のプリミティブによって実現される。

・CONNECT(自分通信端点、相手通信端点の並び)

指定された自分の通信端点と複数の相手通信端点の間に同報コネクションを生成する。もし、すでに自分の通信端点が同報コネクションを持っていたら、このプリミティブで指定された相手が既存の同報コネクションのなかに取り入れられる。

DISCONNECT（自分通信端点、相手通信端点の並び）

指定された自分の通信端点とのあいだに生成されている同報コネクションから、指定された相手を削除する。削除された通信端点は、持っていた情報を捨てることができる。もし、すべての相手が削除されたらコネクションは解放される。

同報コネクションでの送信メッセージは、コネクションの必要なメンバに選択的に送られる。メッセージを、同報コネクションのメンバすべてに送り、受けた側が破棄する方法も考えられるが、不要なメッセージ破棄のための負荷が高くなる。必要なメンバからなる同報コネクションを新たに設定し、コネクションのメンバすべてにおくる方式では、コネクションの設定・解放のオーバヘッドを無視できない。

データ送信のプリミティブでは次のよう通信相手をつねに指定する。

SEND（自分の通信端点、相手の通信端点の並び、送信データ長、送信データ）

以上のマルチキャスト・プリミティブにより、制御は必要最低限のサイトへのパケット送信によって実現できる。この機構は、再送の制御においても同様に働くため、真に再送を必要とするサイトだけへのマルチキャスティングによる再送を可能にした。

3．2　同報通信による問い合わせ処理

分散演算の処理では、二つのサイトの異なるリレーションを逐次的に照会することによってのみ、処理可能な場合がある。このような場合には、従来の同報通信ではまったく利点がでない。このような場合がシステム全体の処理に占める割合は無視できないことが、検討により明らかになった［坂斎88］。

すでに述べたように、通信コストの削減では通信準備時間の低減が重要であるが、同報通信の単純な利用が、そのまま通信準備時間の低減を意味するものではない。同報通信の長所は、複数のサイトへのパケットの同時送達にある。この特質により、通信回数、ひいては通信準備時間の総和の減少が期待できる。しかし、同一のサイトに複数のメッセージを送らねばならない場合、同報通信のこの長所は有効に機能しない。この際に必要なのは、同一サイトへ複数のメッセージを同時に送達する機能だからである。また、分散データベース制御では、各サイトにおくるメッセージがすべて同一という場合はむしろ稀である。各サイトが、ある程度の自律性を保持しながら協調動作する以上、この特質は当然のものである。このため、単なる同報通信の手段を提供したのみでは、分散データベース制御のためには不十分である。

単なる同報通信が有効に機能しない典型的な例を図２に示した。この例では、問い合わせは単純に分解され、ネステッド・ループ法と呼ばれる方式で処理されている。この場合、分散データベースシステムは、問い合わせを複数のサイト上に分散した複数のリレーションに対する一見独立した２つの問い合わせに分解する。図２に示すように、この問い合わせの処理は、個々のリレーションの逐次的なアクセス処理から構成されている。従って、同報通信を利用しても、通信回数はまったく減少しないように見えるのである。

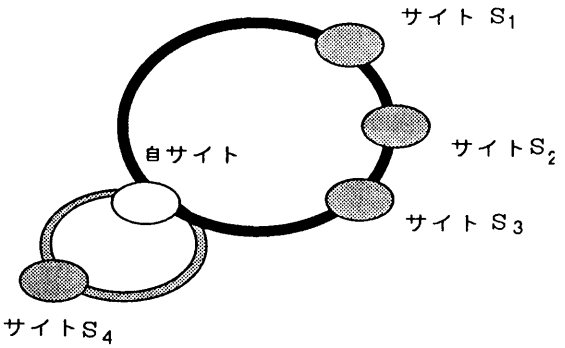


図１（ａ）従来の同報コネクション

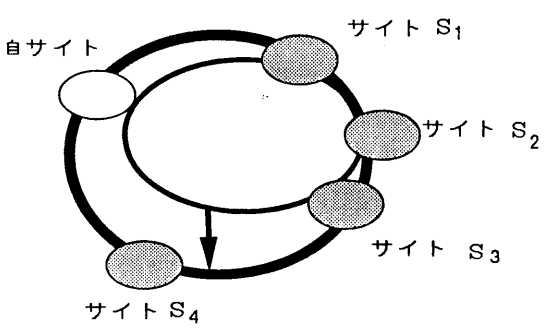


図１（ｂ）動的な同報コネクション

重畳同報方式は、複数サイトへのメッセージ同時送達ばかりではなく、同一サイトへの複数メッセージの同時送達を実現し、通信回数を削減する目的で開発された。図２に示された処理シーケンスは、サイトＳ₂上に位置するリレーションのタプルが尽きた場合、当該サイトへのCLOSE 処理と、サイトＳ₁への次回の FETCH処理とが同時に発行可能であることを示唆している。

このような場合に CLOSEと FETCHを重畳して、Ｓ₁とＳ₂双方のサイトにメッセージを同報する。受け取ったサイトで、それぞれ必要な部分を抜き出し、処理を行うのである。

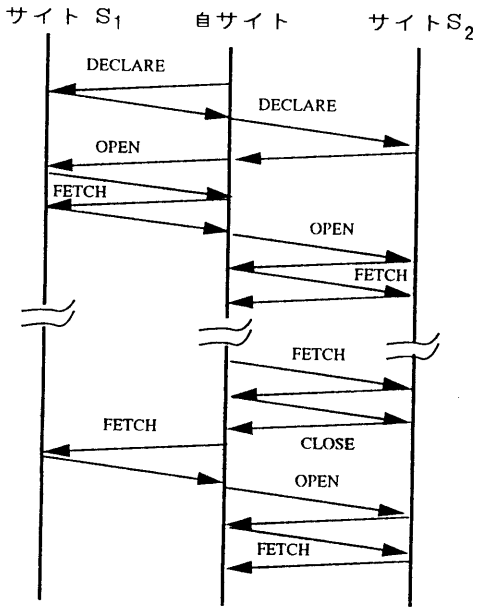


図2　同報通信が機能しない問い合わせ例

重畳同報を実現するプリミティブは、以下のような形式で提供されている。このプリミティブは、従来の同報通信、および重畳通信を統一的に扱うことのできる形式をもっている。以下に示したプリミティブPIGGYMULTIの形式は、本研究での分散データベースシステムでの典型的な重畳同報を実行するものである。

PIGGYMULTI((サイトの並び１）、メッセージ１、（サイトの並び２）、メッセージ２‥‥（サイトの並びＮ）、メッセージＮ）

このプリミティブは、サイトの並び１で指定された複数サイトへメッセージ１を、サイトの並び２で示された複数サイトへメッセージ２を、１回の同報通信で送達する。メッセージを受け取ったサーバは、重畳されたメッセージの中から自サイトへあてたものを取り出す。

重畳同報プリミティブに与えられるメッセージが単一の場合、すなわちプリミティブが以下の形式で記述されるとき、実行結果は従来の同報通信と同じとなる。

PIGGYMULTI((S₁, S₂, S₃), メッセージ１）

逆に、サイトの並びがすべて同一のサイトを指定している場合には従来の重畳と同じ動作になる。この場合のプリミティブの記述は、以下のようになる。

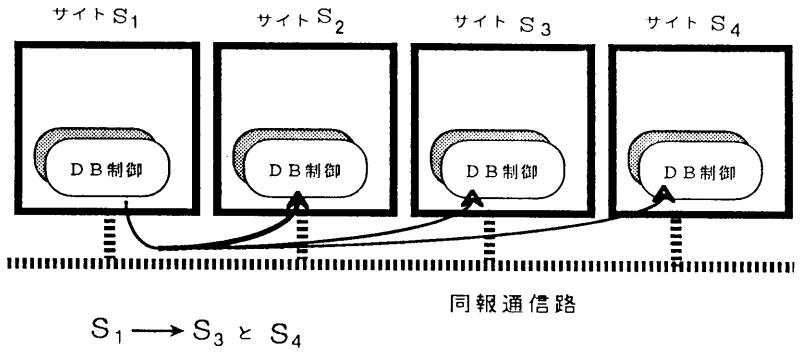
PIGGYMULTI((S₁), メッセージ１, (S₁), メッセージ２, ‥‥)

さらに、この方式は、必要なサイトのみに同報を行うマルチキャスト通信をベースにしている。これにより、重畳同報が一層有効な通信方式となることが、これまでの実装により確認された。

３．３　マルチキャスト指向同報通信

すべてのサイトにブロードキャストパケットを配信する同報通信（以後無条件同報通信と呼ぶ）での問題の解決のために、マルチキャスト通信を実装した。マルチキャスト通信とは、必要なサイトにのみパケットが配送される通信方式である。

無条件同報通信のもたらす問題は、それを必要としないサイトにまでパケットが配送される性質に由来する。図３はこの例を示す。サイトＳ₁がサイトＳ₃とサイトＳ₄にメッセージを同報しようとしてブロードキャストパケットを送り出す。サイトＳ₁とサイトＳ₃には、このメッセージが到着するが、これと同時に、本来そのメッセージを必要としないサイトＳ₂にも、メッセージは届けられる。サイトＳ₂のデータベース制御部は、サイトのＣＰＵ時間を費やして、このメッセージを破棄する必要がある。不必要なブロードキャストパケットの廃棄のために、ＣＰＵの能力が不必要に浪費されることになる。



サイトＳ１から、サイトＳ３とＳ４への同報送信が行なわれているが、サイトＳ２にも、同報パケットが不必要に送られて、Ｓ２のＣＰＵ時間を浪費している。

図３　ブロードキャストパケットによる制御情報伝達

多くのサイトが同時にブロードキャストパケットを送出したとする。当然、受信のためのサイトの負荷は高くなる。ここで示したよりもはるかに大量のブロードキャストパケットが全サイトで受信され、破棄されることになる。中央局から全部の子局への通信といった従来の典型的な同報通信の形態では、ここで述べたような問題はない。しかし、システム管理および分散演算などのデータベース分散制御を行う上では、破壊的な問題を引き起こす。

マルチキャスト通信の実現は、無条件の同報通信が引き起こす問題に対する自明の解答のように思われる。しかし、現実に利用可能な通信媒体は、マルチキャスト通信を提供していない。

本研究で実装したマルチキャスト通信は、同報通信機能を提供するＬＡＮのパケット上にプロトコルとして実現したものである。ベースとした通信媒体が同報通信のみを提供しているため、すでに述べたような問題点を完全に解決できたわけではない。ただし、マルチキャスト通信パケットを処理するプロトコルマシンは、分散データベース管理システムから独立したソフトウェアモジュールとして実現された。配送された同報通信パケットのうち、当該のサイトに必要でないものは、このソフトウェアモジュールで廃棄され、分散データベース管理システムの動作には影響を与えない。

この方式は、以下のような拡張が有効であることを示唆している。１つは、マルチキャスト通信をより低いレベルの通信プロトコルとして実現し、ハードウェアが必要な通信パケットのみを受信するようにするアーキテクチャである。もう１つは、マルチキャスト通信パケットを処理するソフトウェアモジュールを、分散データベース管理システムとは別のプロセサ（通信プロセサ）で実行させるアーキテクチャの採用であり、第一の方式の変形と言ってもよい。逆に言えば、このようなアーキテクチャ上のサポートなしには、同報通信をベースにした分散データベースシステムの効果的な実装は、それほど容易なものではない。

マルチキャスト通信方式の実現は、これを利用したシステムモニタリングを現実に近づけた。同報通信を利用したシステムモニタリングとは、システム内で交換されるブロードキャストパケットをモニタすることで、システムの実時間の動作を監視しようとするものである。プログラムの中にモニタリングのためのメカニズムを埋込むと、モニタリングのための動作がオーバヘッドを生じ、システムの動作、特にパフォーマンスに悪影響を及ぼす可能性がある。分散データベースシステムは、問い合わせ、または更新処理の実現のためにメッセージ交換を行う。同報通信の特質を生かして、これらのメッセージをモニタし、システムの動作を監視しようとする考えは極めて魅力的である。しかし、すでに述べたように無条件の同報通信では、交換されるメッセージの量は爆発的なものとなる可能性がある。システムがより巨大化したとき、同報通信を利用してのシステムモニタリングは現実性を失う。

マルチキャスト通信は、爆発的なメッセージ交換という問題を解決し、真に必要なメッセージのみをモニタすることを可能にする。本研究では特に、更新処理に不可避のグローバルデッドロック問題を、マルチキャスト通信をベースとしたモニタリングによって解決する方法に関する検討を行っている。

４．　分散更新処理

著者らは、分散更新処理と、それに伴うグローバルデッドロックの問題の、マルチキャスト指向の同報通信による解決を試みている。グローバルデッドロックとは、分散データベースを構成する複数のサイト上のトランザクション間にまたがるデッドロック状態を意味する。マルチキャスト指向の同報通信の持つ即時性と同報性が、グローバルデッドロック問題における技術上の問題点に解決を与える可能性がある。本研究で実装した同報コネクションのメカニズムは、サイトから送信されるパケットの高信頼性を保証する。かつ、システム内の複数サイトに同時送達される、同報通信の特長も維持されている。

更新処理に伴うデータの一貫性保証は、集中型データベースシステムでも煩雑な問題である。多くの集中型システムではロック（施錠）方式、またはタイムスタンプ（時刻印）方式が採用されている。ロック方式とは、複数の処理プロセス（トランザクション）から更新されるデータにロックをかけ、同時更新による一貫性の破壊を防ぐ方式である。一方、タイムスタンプ方式とは、トランザクションにタイムスタンプを与え、このタイムスタンプによって矛盾のない同時更新を保証しようとするものである。

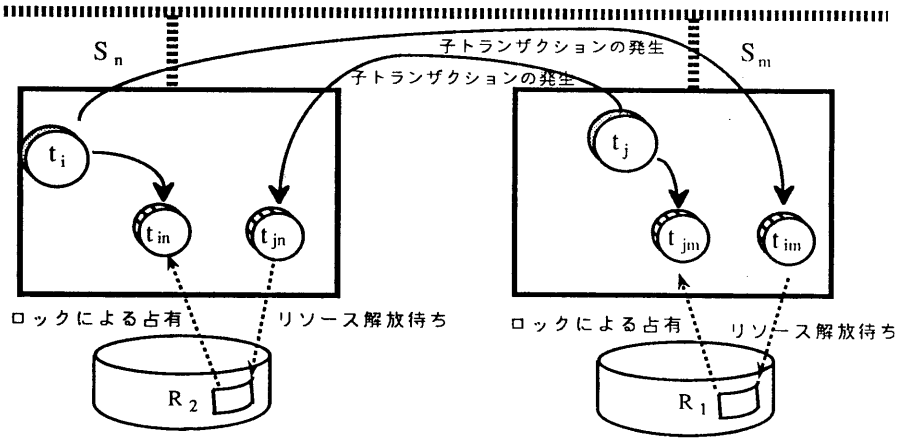
４．１　グローバルデッドロック

ロック方式を分散データベースにそのまま適用した場合、グローバルデッドロックが避けがたい問題となってくる。図４に示すように、グローバルデッドロックは、各サイト上のローカルなデータベース管理システムでは解決困難な問題を引き起こす。サイトＳn 上のトランザクションｔi は、遠隔のサイトＳm 上のリソース（リレーショナルデータベースで言えば、表・タプルなど。以後、リソース）Ｒi に対する処理を行なうために、子トランザクションｔimを発生させる。一方、サイトＳn 上のトランザクションｔj も、遠隔サイトＳn 上のリソースＲ2 への処理を行なうため、子トランザクションｔjnを発生させるだろう。このとき、リソースＲ1 とＲ2 が、それぞれトランザクションｔj とｔi の子トランザクションｔjmとｔinによってすでにロックをかけられていたとする。ｔimはｔjm の、ｔjnはｔinのリソース解放を、それぞれ待ち合わせる（競合状態）。図４に示すとおり、この状況はｔi とｔj の間のデッドロックである。しかし、サイトＳn とＳm 上のローカルなＤＢＭＳは、デッドロックの発生を認識できない。

問題を引き起こすのは、デッドロックを制御（予防、回避、検出）するための制御情報の交換にともなう通信コスト、および通信遅延である。既に述べたようにローカルなＤＢＭＳは、図４のような分散型のデッドロック（グローバルデッドロック）の発生を認識できない。ロック制御に伴う、競合状態の情報を交換し、デッドロックの制御を実現するメカニズムが必要となる。分散データベース管理システムの実現上、情報交換に伴う通信コストが問題となるわけである。[KNAP87]によると、１トランザクションが最大１つのリソースに対して競合し得る処理モデル（Single Resource Mode。以後、単一リソースモデル）でのデッドロック検出には、（関連するサイト数）^{2} の通信回数を必要とする。通信回数が、通信コストの制御要因であるとすれば、デッドロック制御にともなって増加する通信回数は大きな問題をもたらす可能性がある。

同時に、デッドロック制御に必要な通信処理は、通信コストだけではなく、通信遅延やパケットの紛失の問題を引き起こす。通信遅延やパケットの紛失が引き起こすのは、存在しないデッドロックを検出してしまう、幽霊（ファントム）デッドロックと言われる問題である[Wuu85]。

著者らの方式は、競合状態をモニタし、グローバルデッドロックを検出・解消しようとするものである。すなわち、システム内にグローバルデッドロックの発生を検出するサイト（またはプロセス）を置くものとする。ロック制御を行うための交換メッセージに、デッドロックに関する制御情報を重畳同報機能によって、このモニタに対して通知するのである。本研究で実装したマルチキャスト指向の同報通信メカニズムは、競合状態のモニタの効率的な実現をサポートした。同時に、無条件の同報通信が引き起こす問題を解決している。



サイトＳm上では、子トランザクションｔｉｍが、子トランザクションｔｊｍを待ち合わせる。一方、サイトＳｎ上では、ｔｊｎがｔｉｎを待ち合わせる。システム全体では、ｔｉとｔｊの間のデッドロックであるが、サイトＳｎとＳm上のローカルなＤＢＭＳにとっては、ｔｊｎとｔｉｎ、ｔｉｍとｔｊｍの間の単なる競合状態に過ぎない。

図４　分散データベースシステムにおけるグローバルデッドロックの発生

同報通信の特長である同報性と即時性は、通信回数の低減による通信コストを削減し、通信遅延の減少にともなうファントム・デッドロックの可能性を引き下げる。また、本研究で実装したマルチキャスト指向の通信メカニズムが、パケット紛失の可能性を著しく低いものとした。

もちろん、同報通信を実現しただけでは、グローバルデッドロックに効率的に対処できるわけではない。リソースの競合状態をモニタする必要が生じるケースに関する詳細な検討を行った結果として得られたのが、以下に述べるような結論である［小坂89］。

4．2　モニタリングアプローチの検討

グローバルデッドロックを引き起こす可能性のある2つの場合に関して詳細に検討する。S_i 上のトランザクション t_p が S_j 上のトランザクション t_q にブロックされ、$t_p \rightarrow t_q$ のサイト間競合が生じたとする。

Case.1：　t_p が、他のサイトのトランザクションによって待ち合わされていない。

Case.2：　t_p が、他のサイトのトランザクションによって待ちあわされている。

（1）　Case.1の詳細検討

Case.1は、以下のようなケースに分類される。

(1)　$t_p \rightarrow t_q$ のサイト間競合が生じる前に、t_q が、同じサイト上の他のトランザクションを待ち合わせていたとする。この場合、$t_p \rightarrow t_q$のサイト間競合がグローバルデッドロックを発生させる可能性があるとして、モニタに通知される必要がある。

(2)　$t_p \rightarrow t_q$ のサイト間競合状態の発生が先行し、その後 t_q が同じサイト上の他のトランザクションを待ち合わせたとする。t_q を待ち合わさせているトランザクションが、他のサイト上のトランザクションとの間に競合状態を生じた場合も、グローバルデッドロックを生じる可能性がある。

(3)　$t_p \rightarrow t_q$ のサイト間競合が生じる前に、t_q が、他のサイト上のトランザクションを待ち合わせていたとする。この場合、$t_p \rightarrow t_q$のサイト間競合がグローバルデッドロックを発生させる可能性がある。

（2）　Case.2の詳細検討

以下の2つの場合に分類される。

(1)　t_q が、同じサイト上の他のトランザクションを待ち合わせている。この場合、グローバルデッドロックが生じる可能性がある。

(2)　t_q が、同じサイト上の他のトランザクションを待ち合わせていない。t_q が、このあと他のトランザクションと競合状態を引き起こし、結果的にグローバルデッドロックを引き起こす可能性がある。ただし、t_p から t_q へのサイト間競合状態が生じた時点では、グローバルデッドロック発生の有無が判断できない。

以上のような検討から、リソースの競合状態が生じた場合に、モニタサイトにグローバルデッドロック発生の可能性を判断する規則は、以下のようになる。

あるサイト S_n で、トランザクション t_i が、トランザクション t_j がロックしているリソース R_i に対して競合状態になったとする。もし、トランザクション t_i 自身がサイト S_n 以外で発生したトランザクションであるか、または t_i に対して待ち合わせ状態になっているトランザクションのなかに、サイト S_n 以外で発生したトランザクションが存在し、かつ t_j 自身が、サイト S_n 以外で発生したトランザクションであるか、または t_j が待ち合わせているトランザクションのなかに、サイト S_n 以外で発生したトランザクションが存在したとする。このとき、トランザクション t_i と t_j の間の競合は、グローバルデッドロックを発生させる可能性がある。

5．　将来の課題

以上、述べてきたように、著者らは、分散データベース管理システム実現上の技術的な問題を、同報通信の機能を最大限に利用することで解決を図ってきた。検索処理に関連する問題については、前期においてかなりの前進を見ることができた。

残された課題として、分散システムの利点である並列処理の、同報通信による効果的な実現がある。この点に関しては、本稿で触れることができなかった。また、分散更新処理の検討は、まだ緒についたばかりである。4章では限定された更新モデルのみを論じた。今後は、データの重複を考慮して、分散更新処理の実現を検討しなければならない。

ファントム・デッドロックも解決されなければならない、分散データベース特有の問題である。この問題は、通信コスト・通信遅延によって引き起こされる。同時に、分散システムの特徴の1つである、各サイトの自律性も、同じような問題を引き起こす可能性をもっている。

6．　終わりに

本稿では、分散データベース管理システム実現のための課題を概観したのち、本研究でのアプローチを、通信アーキテクチャ、プロトコル、問い合わせ処理に焦点をあて、その有効性を論じた。さらに、分散更新

処理における問題点、およびグローバルデッドロック問題と解決への同報指向アプローチを述べた。今後、将来への課題として指摘した問題点を中心に、同報通信指向アプローチによる分散データベースシステムの研究開発を進める予定である。

本研究開発は、通商産業省工業技術院大型プロジェクト「電子計算機相互運用データベースシステムの研究開発」の一環として、新エネルギー・産業技術総合開発機構（ＮＥＤＯ）より委託を受けて実施したものである。

Reference

[BHID88] Bhide, A., "An Analysis Three Transaction Processing Architecture.", In Proceedings of the 14th VLDB Conference, Los Angeles, 1988

[CHAN83] Chang, J. M., "LAMBDA: A Distributed Database System for Local Area Network.", Database Engineering, IEEE, Vol. 4, 1985

[HEVN79] Hevner, A. R. and Yao, S. B., "Query Processing in Distributed Database Systems.", IEEE Transaction on Software Engineering, Vol. SE-5, No. 3, May 1979

[KNAP87] Knapp, E., "Deadlock Detection in Distributed Database.", ACM Computing Surveys., Vol. 19, No. 4, December 1987

[KOTE89] 小寺、坂本、川上、正田、"モニタリングによるグローバルデッドロック検出へのアプローチ"、情報処理学会第３８回全国大会予稿集（II）２Ｑ－６、1989年３月

[坂斎88] 坂本、斎藤、正田、"同報通信を指向した分散ＤＢ"、情報処理学会データベースシステム研究会 68-6、1988年11月

[WUU85] Wuu, G. T, and Bernstein A. J., "False Deadlock Detection in Distributed Systems.", IEEE Transaction on Software Engineering, Vol. SE-11, No.8 August 1985