データベース・システム 73－5
オペレーティング・システム 44－5
（1989. 9. 8）

分散セグメンテーションに基づくデータベースアクセス方式

國枝 和雄　水谷 高康　大久保 英嗣　津田 孝夫

京都大学工学部情報工学科

分散データベース向きのデータアクセス方式として分散セグメンテーションを提案する。分散セグメンテーションによって、ユーザは、主記憶と2次記憶といった記憶階層の管理から解放される。さらに、各データの存在位置を知ることなく、名前によってデータをアクセスすることが可能となる。我々は、現在開発中のデータベースオペレーティングシステムμOPT-Rにおいて本方式を実現し、その有効性を確認した。本稿では、分散セグメンテーションの実現方式と、それに基づくデータベースアクセス方式について述べる。

Database Access Method based on the Distributed Segmentation

Kazuo Kunieda, Takayasu Mizutani, Eiji Okubo and Takao Tsuda

Department of Information Science, Faculty of Engineering, Kyoto University

We propose a data access method which is suitable for distributed database processings. We call it the distributed segmentation. The distributed segmentation release users from the management of the memory hierarchy between main memory and secondary storages. Furthermore it make possible for users to access data without specifying the data locations. We have implemented this access method in the database operating system μOPT-R, and confirmed its effectiveness. This paper presents the technique for implementing the distributed segmentation, and the database access method based on it.

1. はじめに

分散データベースの形態としては、WANによる結合でバッチ的な処理を行うことを基本とするものと、LANによる結合でユーザからの問い合わせに対してリアルタイム性を重視するものとが考えられる。我々は、今後より要求が高まると思われる後者の形態に着目し、その処理速度向上を目的として、メモリ管理、データ管理、入出力などをデータベース処理向きに設計したデータベースOS μOPT-Rの開発を行っている[1][2]。データベースOSにおいては、OSの各機能をデータベース専用に設計可能であることに加え、OSの提供する物理的実体に対する管理機構とDBMSにおいて実現される論理的実体に対する管理機構との不整合や冗長性などといった、従来型データベースの問題点[3]を解決することが期待できる。

本稿では、μOPT-Rのデータ管理方式として用いた分散セグメンテーション方式の概要と、それに基づくデータベースアクセス方式について述べる。

2. セグメンテーション

μOPT-Rのデータ管理方式として分散セグメンテーションを用いた背景について述べる。

2.1 利点

通常、OSにおけるデータの物理的な管理単位はファイルであり、データの安全性制御や一貫性制御もファイルを単位として行われている。このファイルという実体は、データが2次記憶上に格納されている時点でのものであり、一旦アプリケーションプログラムによってデータが主記憶上に読み込まれるとデータとファイルは独立なものとなる。しかも、OSからデータに対する安全性や一貫性の制御を行うことは不可能となる。そこで、データに対してこのような制御を行うためには、データを要求しているアプリケーションプログラムが独自に制御機構を設ける必要がある。一方、主記憶および2次記憶を含めた仮想記憶空間において、ある意味を持つ情報をまとめ、それを単位として記憶空間の管理を行う方式がセグメンテーションであり、その論理的単位をセグメントと呼ぶ。セグメントは、それが主記憶上と2次記憶上のどちらに存在しても、OSのメモリ管理によって制御することが可能である。したがって、データベースにおけるように、データの管理方式を限定可能な場合には、非常に有効な方法であると考えられる。特に、マルチタスクあるいはマルチプロセッサシステムにおいて、演算間および演算内の並列性を検出して並列に処理を行う場合には、演算タスク間で発生するアクセス競合を処理する機構が必要となるが、そういった機構を完全にOS内部において実現できるセグメンテーションは有効である。データベースシステムにおいてセグメンテーションを行う利点としては以下のものが挙げられる。

(1) 演算を行う各モジュールは、他のモジュールのデータの使用状況を考慮することなくデータを要求することが可能である。

(2) 2次記憶上のみならず主記憶上に存在するデータの共有も容易である。

(3) データに対するアクセス要求は、データベースにおける論理的実体への要求としてOSへ渡されるため、OS層においてデータベースのコンテクストに基づいたアクセス制御が可能である。

(4) 識別子から物理アドレスへのマッピングが直接行われる。従来のDBMSでは、DBMSにおける(データ識別子→ファイル識別子)、OSにおける(ファイル識別子→物理アドレス)の最低2度のマッピングが行われる。

2.2 分散セグメンテーション

我々は、分散環境においても前述の利点が得られると考え、μOPT-Rにおいて分散セグメンテーションを導入した。分散セグメンテーションは、セグメンテーションによって管理される空間を分散システム全体に拡大し、異なるサイトの主記憶と2次記憶を一元的に管理する方式である。これによって、従来型の分散データベースにおいてOS層とDBMS層に渡って管理されていたシステム構成の情報を、完全にOS層内部に隠蔽し、DBMS層に対してセグメントの透過性を提供することが可能となる。また、データの共有度が向上することにより、各サイト間で発生する入出力や通信量が低減すると考えられる。

3. システム構成

μOPT-Rで対象とするシステムは、複数のサイトが多重アクセスバス方式の単一のLANによって結合された分散システムである。各サイトは、単一のプロセッサ、ローカルメモリ、ローカルディスク、通信装置及びその入出力装置より構成される。したがって、サイト間にはメモリおよびクロックの共有はなく、サイト間通信はメッセージ交換によってのみ可能となる。

μOPT-Rの現在の実験システムは、サイトにPC-9801VM、LANにBRANCH4800(共にNEC 製)を用いている。また、システムの記述言語としてはC言語と一部アセンブリ言語を使用している。

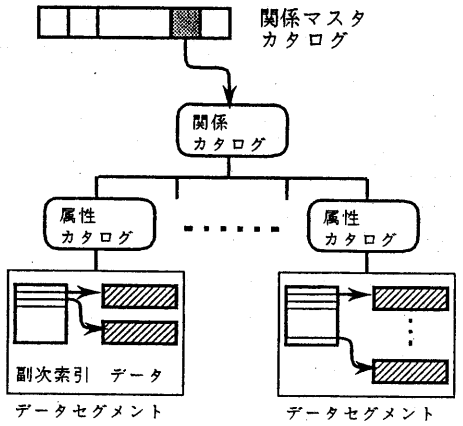

図1 セグメント階層

表1 セグメント識別子

セグメント種別	識別子の構成要素
関係カタログ	所有者、関係名
属性カタログ	所有者、関係名、属性名
データセグメント	所有者、関係名、属性名、クラスタ番号

4. データ構造

4.1 セグメント種別

μOPT-Rにおけるデータベースの論理構造は"関係"であり、"関係"の物理的構成要素はセグメントである。セグメントは、メタデータを格納するカタログセグメントと、データそのものを格納するデータセグメントに大別される。これらのセグメントは、木構造を形成しており、一つの木は一つの"関係"を表現し、データベース全体では林構造となる(図1参照)。

(1)カタログセグメント

カタログセグメントは、関係マスタ、ビューマスタ、ドメインマスタ、関係、属性、ビュー、ドメイン、グラントの各カタログを構成している。

(2)データセグメント

データは、転置型ファイル形式で格納されている。さらに、各データセグメントはクラスタ化されており、それら各々に副次索引が付加されている。

4.2 データの配置方式

μOPT-Rにおける第一の目標は処理速度の向上である。そのために、通信コストをできる限り小さくするようにデータを配置している。

(1)データを分散する単位

μOPT-Rにおけるデータ分散の最小単位は"関係"である。タプルや"属性"など、さらに細かい単位での分散配置も考えられるが、セグメント間の関係に基づくデータへのアクセス制御が複雑になる点、およびサイト間の通信量が増加する点で不利となる。

(2)データの二重化

データの二重化に関しては、サイト内の2次記憶装置の二重化や、複数サイト間での同一データの二重化などが考えられる。これに関しては、今後サポートする予定である。

4.3 セグメント識別子

分散システムで用いる識別子に関して問題となるのは、識別子と物理アドレスとの対応付けである。一般に、透過性の実現された識別子であっても、識別子と物理アドレスがまったく独立している場合と、識別子中に何らかの位置情報を含む場合がある。前者は、完全にデータの存在する位置と識別子が独立で柔軟性も高いが、識別子参照時のオーバヘッドは大きくなる。一方、後者は、サイトを識別するための値がその識別子中に埋め込まれているような場合であり、識別子参照時の処理は比較的容易である。

μOPT-Rにおけるセグメント識別子は、セグメントの種別および内容に依存する論理的な要素のみで構成され、セグメントのロケーション情報とは完全に独立している(表1参照)。これは、データベースにおいては、識別子情報を含むメタデータ自身もデータであることによる。すなわち、システム構成とメタデータの独立性を高めるためには、ロケーション情報と独立した識別子が必要となる。

5. セグメント管理部の概要

5.1 構成

μOPT-Rのセグメント管理部の構成を図2に示す。セグメント管理部は、以下の3つのモジュールで構成されている。

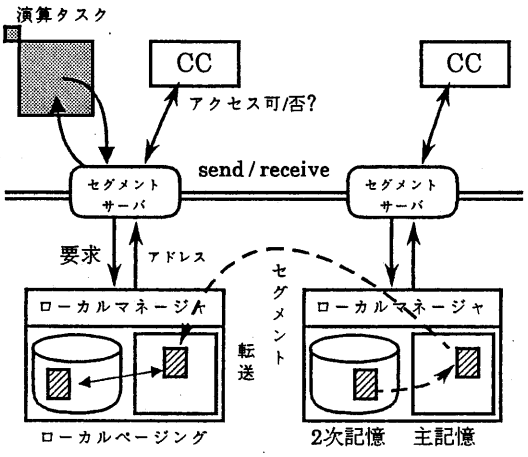

図2 セグメント管理部の概要

(1)ローカルマネージャ(LM)

各サイトにおいて、主記憶と2次記憶間でのセグメント入出力を管理するモジュールである。セグメントはページ(1024バイト)を単位として管理され、LM内のページング機構によって、随時2次記憶と主記憶間で転送される。LMは、SSからのセグメント要求によって、セグメントを主記憶上に配置し先頭アドレスを返す。

(2)セグメントサーバ(SS)

各サイトのサーバ間で相互にセグメント情報およびセグメント要求を通信することによって、セグメントの透過性を実現する。セグメントに関する分散処理機能は、このSSによって提供される。

(3)同時実行制御部(CC)

μOPT-Rでは、同時実行制御機能をセグメント管理部に組込むことによって、セグメントアクセスと同時実行制御に関する処理を統一して行っている。また、この部分を変更することによって、異なる同時実行制御アルゴリズムを用いることが可能である。現在、μOPT-Rでは2相ロックに基づく方式を用いている。

5.2 処理方式

5.2.1 セグメント管理部の処理手順

演算タスクにおいて発行されたセグメント要求は、セグメント管理部では以下の手順で処理される。

(1) セグメント要求がSSに送られる。

(2) SSは、CCに対して要求を処理可能か否か問い合わせ、不可の場合には不当要求であることを演算タスクに返信する。

(3) SSは、カタログを調べセグメントがローカルサイトに存在するか否か調べる。

ローカルサイトに存在する場合:

(4a) 要求をLMに送る。

(5a) LMはセグメントを主記憶上に転送し、その先頭アドレスをSSに返す。

(6a) SSは、得られた先頭アドレスを演算タスクに返す。

リモートサイトに存在する場合:

(4b) 要求をリモートサイトのSSに対して一斉に送信し、返信を待つ。

(5b) リモート側のSSの中で当該セグメントが存在するサイトのSSをSSR、要求元サイトのSSをSSLとすると、SSRは、(4a)~(5a)の手順でセグメントの先頭アドレスを獲得する。

(6b) SSRは、SSLに対してセグメントの内容を転送する。

(7b) SSLは主記憶上に領域を確保し、SSRから受け取ったセグメントの内容をその領域へ転送する。

(8b) SSLは、領域の先頭アドレスを演算タスクへ返す。

5.2.2 各セグメント要求に対する処理

セグメントサーバが提供するセグメントへのアクセス機能は、セグメントの生成/削除/参照/更新/拡張/仮解放の各要求に対する機能である。本節では、これらの各要求に対する処理の説明を行う。

○参照/更新/拡張要求に対する処理

(1) 演算タスクはOSに対して参照/更新/拡張したいセグメントを要求する。

(2) OSがセグメントを主記憶上に配置し、その先頭アドレス(物理アドレス)を演算タスクへ返す(5.2.1参照)。

(3) 演算タスク内では、得られた先頭アドレスとセグメント内オフセットを用いてセグメントの内容を参照/変更する。

(4) 処理が終了すると、OSに対してセグメントの処理終了を通知する。

以上は一連の処理であり、その間セグメントの実体は主記憶上の固定された位置に存在することが保証される。

参照の場合、共有されたセグメントへのアクセスを高速化するために、一つのセグメントの複製は各

サイトには高々一つ、かつ複数のサイト上に多重に存在することを許している。したがって、要求元のサイト上にセグメントの複製が存在すれば、ネットワークを介したセグメントの転送は行われない。

変更を伴う要求(更新/拡張)の場合、セグメントの複製を主記憶上に作ることは許されず、セグメントは一つの実体のみが存在する。これは、データの正当性を保証するためのものである。したがって、複数のサイトから1つのセグメントに対して同時に更新要求が行われた場合、たとえ同一トランザクション内からの要求であっても認められない。

○削除要求に対する処理

トランザクションの後退復帰に備えて、削除要求のあったセグメントは実体を残したまま削除状態に遷移させる。そして、トランザクションの終了処理を行うときに実体を削除する。トランザクションがコミットされた時点で、セグメントはトランザクション発行前の解放状態に戻される。

○生成要求に対する処理

μOPT-Rでは、一つの"関係"を構成するセグメントは同一サイトに配置することになっている。したがって、セグメントの生成は、そのセグメントが属する"関係"が存在するサイトにおいて行われる。"関係"が存在せずに、その生成要求によって新たな"関係"が作られる場合には、セグメントは要求元サイト上に生成される。

すでに存在するセグメントに対する生成要求は不当であるが、当該セグメントが同一トランザクションからの削除要求によってすでに削除状態に遷移している場合には正当な生成要求である。したがって、その場合には、セグメントを削除状態から専有状態に戻し、さらに、内容を初期化してセグメントを新たに作成したものとする。

○仮解放要求に対する処理

演算タスクのセグメントへのアクセスは、一旦セグメント要求によってOSから先頭アドレスを得ると、それ以降はセグメント内オフセットを用いてポインタで自由にアクセスすることが可能である。したがって、セグメントは、トランザクションがコミットされるまではページアウト不可能である。しかし、トランザクション内で多くのセグメントを使用している場合には、主記憶領域の不足によって、トランザクションの実行を行えない可能性がある。

これを避けるために、それ以降ポインタによるアクセスを行わないという条件の元で、トランザクション内部でもページアウト可能とする宣言が仮解放である。仮解放されたセグメントに再びアクセスする場合には、OSに対して再度セグメントの更新/参照要求を発行しなければならない。

演算タスクにおいては、更新/参照要求と仮解放要求を繰り返すことによって、主記憶を有効に利用できる。

5.2.3 セグメントの状態と遷移

セグメントは、5.2.2で述べた各要求によって以下の6種類の状態を遷移する(図3参照)。

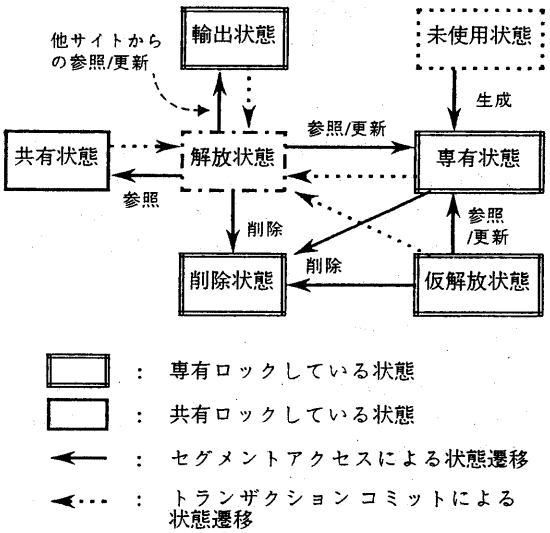

図3 セグメントの状態遷移

(1)解放状態

セグメントが現在どのトランザクションからも使用されていない状態である。そのセグメントを使用しているトランザクションをコミットすることによって、この状態へ遷移する。この状態から他の状態に遷移するためには、要求発行元トランザクションはセグメントをあらかじめロックしておかなければならない。

(2)専有状態

セグメントが、ただ一つのトランザクションから専有され、主記憶上に置かれている状態である。セグメントを専有ロックしているトランザクションからの更新/参照要求が発行された場合に、この状態へ遷移する。

(3)共有状態

セグメントを共有ロックしているトランザクションから参照要求が発行された場合に遷移する状態であ

り、この状態のセグメントは、複数のトランザクションから共有することが可能である。

(4)仮解放状態

セグメントが他のセグメント要求を要因としてページアウトされることを許す状態である。専有状態にあるセグメントに対して、仮解放要求が発行された場合にセグメントはこの状態へ遷移する。

(5)削除状態

専有状態にあるセグメントがトランザクションからの削除要求によって遷移する状態である。セグメントの実体は、主記憶上からは削除されるが、トランザクションの後退復帰に備えて2次記憶上に残される。

(6)輸出状態

自サイトの2次記憶にあるセグメントを、他サイトからの要求によって要求元サイトの2次記憶へ輸出している状態である。トランザクションの終了時には、輸出先サイトのセグメントの内容が、輸入元サイトの2次記憶上に反映される。

5.2.4 同時実行制御

現在μOPT-Rでは、2相ロック方式で同時実行制御を行っている。以下では、この方式に基づくμOPT-Rの同時実行制御について述べる。

○トランザクションの開始処理

2相ロック方式では、トランザクション内でセグメントアクセスが行われる場合には、それに先だってセグメントをロックしなければならない。ここで、μOPT-Rにおいて属性値の格納されるデータセグメントは、入出力オーバヘッドを低減するために、分割･クラスタ化されている。したがって、セグメントを単位としたロッキングではロック対象が多くなり負荷が大きい。そこで、"関係"を単位としたロッキング(専有ロック/共有ロック)を行っている。

また、一般に2相ロック方式ではデッドロックの発生が問題となるが、ロッキングフェーズの処理を不可分に行うことによってこれを回避している。

○トランザクションのコミットメント

トランザクション内のセグメントアクセスが正常に終了した後、トランザクションをコミットするために、以下の3つの処理を行う。

(1)削除状態にあるセグメントの削除

削除状態にあるセグメントは、トランザクションの終了とともに主記憶および2次記憶から削除される。

(2)主記憶と2次記憶の同期

トランザクション中で、変更の行われたセグメントが輸出および輸入状態でない場合には、そのまま、2次記憶に内容を反映させ、セグメントを解放状態に遷移させる。サイト間でセグメントの輸出入が行われている場合には、輸入元および輸出先でそれぞれ次の処理が行われる。輸入元では、更新されたデータの内容を受け取り、それを2次記憶へ反映させる。その後、セグメントを解放状態へ遷移させる。輸出先では、セグメントの内容を輸入元サイトへ送信し、セグメントを削除する。

(3)ロック解除処理

トランザクション終了時には、そのトランザクションによってロックされているすべての"関係"に対して、ロック解除処理を行わなければならない。ロックの種別が共有ロックであり、他トランザクションからもロックされている場合には、ロックは解除されない。ロックの解除後、これらの"関係"の解放を待って中断しているトランザクションを再開させる。

5.2.5 主記憶管理方式

μOPT-Rでは、主記憶を1024バイト単位のページに分割して管理している。主記憶の割り当てが要求された場合、主記憶管理部では、以下の各状態にあるページをその順で走査する。

状態1:現在使われていないページ
状態2:セグメントは存在するが、セグメントの内容をページアウトする必要のないページ
状態3:セグメントが存在し、ページアウトの必要があるページ

また、ページ置き換え方式としては、単純なFirst-Fit方式を用いている。First-Fitで処理を行うと、ページング対象となるページが主記憶の前半部に集中する恐れがあるので、ページ走査時に、前回置き換えを行った領域の次に位置するページから走査することにより、主記憶全域を均等に利用している。また、この方式では、前回置き換えられたページは、次に置き換えの対象となる確率が最も低いという性質が得られる。

セグメンテーション方式では、一般に外部フラグメンテーションが問題となる。この点、μOPT-Rでは、サイズの大きくなる可能性のあるデータセグメントをクラスタ化することによって、すべてのセグメントのサイズを数ページ以内に抑えているため外部フラグメンテーションは問題とはならない。

5.2.6 ローカルページング

分散システムの場合、通信コストが大きな問題となる。したがって、分散セグメンテーションにおいて、リモートサイト上のセグメントを輸入している場合に、ネットワークを介したページングを行うと、システムの効率がかなり低下することが予想される。

そこで、μOPT-Rではトランザクション実行中には、ネットワークを介したページグを行わないローカルページング方式を用いている。この方式では、トランザクション実行中に仮解放状態にあるセグメントに対して発生したページングでは、セグメントを書き出す2次記憶としてローカルサイトの2次記憶を利用している。これによって、トランザクション実行中のネットワークを介したページングを避けることが可能である。

6. おわりに

本稿では、我々の開発しているμOPT-Rにおけるデータ管理方式の概要について述べた。データ管理部はすでに完成しており、実験システム上でのテストおよび性能測定を行った。測定結果は、分散セグメンテーションの有効性を十分確認できるものであった。しかし、実験システムのハードウェアによる制約から、通信速度など絶対的な性能として十分とは言えない部分もあり、今後の改善が必要である。また、本実験システムは、仮想記憶をサポートするハードウェアを備えていないため、ページングに至るすべての処理をソフトウェアで行っており、メモリの保護機構などデータベースOSとして不十分な点が多い。この点に関しては、80386搭載マシンへの移行について検討中である。

なお、本研究は一部文部省科学研究費(課題番号01580025)による。

(参考文献)

[1] 國枝, 水谷, 大久保, 津田: データベース専用オペレーティングシステムμOPT-Rの分散環境におけるセグメント管理方式, 第37回情処全大, 5P-9 (1988).

[2] 水谷, 國枝, 大久保, 津田: データベース専用オペレーティングシステムμOPT-Rにおける演算処理方式について, 第37回情処全大, 5P-8 (1988).

[3] M. Stonebraker: Operating System Support for Database Management, CACM, Vol. 24, No. 7, 412 - 418 (1981).