データベースオペレーティングシステム μOPT-R における分散セグメンテーション方式†

國枝和雄†† 大久保英嗣†† 津田孝夫††

本論文では，分散データベース処理向きのオペレーティングシステム μOPT-R における記憶管理およびデータ管理の方式として，分散セグメンテーションを提案する．分散データベースでは，データのネットワーク透過性や分散同時実行制御を実現するために，大量の入出力やネットワーク通信が発生する．これは，分散データベースにおける性能低下の一因となっている．我々はこの問題を解決し，分散データベースの性能を向上することを目的として，分散セグメンテーションを開発した．分散セグメンテーションでは，各サイトの主記憶と2次記憶の空間をシステムワイドな一つの記憶空間に統合し，その空間上で，データを格納したセグメントを管理する．この記憶階層管理の統合によって，(1)ファイルシステムを介さない高速なデータ入出力，(2)分散環境におけるデータ配置と転送の効率的スケジューリング，(3)ネットワーク透過なデータアクセスなどが可能となる．本論文では，以上の利点を持った分散セグメンテーションの実現方式について述べ，さらに，実験システム上での性能評価の結果を示す．

1. はじめに

従来，データベース管理システム (DBMS) は，汎用オペレーティングシステム (OS) 上の一つのアプリケーションプログラムとして構築されてきた．これらの DBMS では，OS の提供する機能を用いてデータの入出力や主記憶領域の管理を行っており，必ずしもデータベースに適した方式で処理が行われていない[1)~4)]．そこで，我々は，データベースの処理速度の向上を目的として，主記憶管理，入出力管理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[5)~8)]．μOPT-R において対象とするシステム形態は，ローカルエリアネットワーク (LAN) によって結合された，リレーショナル型の分散データベースシステムである．

本論文では，μOPT-R における記憶管理およびデータ管理方式として，分散セグメンテーション方式を提案する．本方式では以下を主な目的とする．

(1) ファイルシステムを介さない高速なデータ入出力
(2) 分散環境におけるデータ配置と転送の効率的スケジューリング
(3) セグメントのネットワーク透過性 (network transparency) の実現

分散セグメンテーションでは，データベースのデータを論理的な単位でセグメントに格納し，主記憶あるいは2次記憶上に配置する．データのアクセスは，セグメント識別子とセグメント内オフセットアドレスを用いて行われる．また，各セグメントがデータベースの論理的な実体に対応しているので，セグメントについてのアクセス制御が，そのままデータへのアクセス制御となる．したがって，従来の DBMS で行われるファイルの入出力操作や，ファイルとデータについての2段階のアクセス制御は行われない．さらに，セグメントのシステムワイドな管理や分散同時実行制御機能によって，ネットワーク透過なデータベースアクセスを可能としている．

以下，本論文では，まず，分散セグメンテーションを実現するに至った背景について述べる．次に，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．最後に，実験結果を示し本方式の有効性について議論する．

2. 概　要

2.1 背　景

一般に，OS におけるデータの物理的な管理単位はファイルである．ファイルアクセスを効率的に行うために，OS ではある決まった戦略が用いられる．例えば，UNIX においては，データ参照の局所性に基づくLRU (Least Recently Used) 方式のバッファプール管理が行われている[9)]．しかし，OS における戦略が必ずしもすべての場合に適切であるとは限らない．DBMS では，大規模なデータを逐次参照する場合が多く[10),11)]，参照の局所性に基づく戦略が有効に作用

† Distributed Segmentation in Database Operating System μOPT-R by KAZUO KUNIEDA, EIJI OKUBO and TAKAO TSUDA (Department of Information Science, Faculty of Engineering, Kyoto University)

†† 京都大学工学部情報工学科

しない．また，ファイルの内容や処理のコンテクストに依存した，複雑なデータアクセス制御や一貫性制御も必要となる[2]．これらの点に対処し入出力の高速化を達成するためには，DBMS において，OS の機能に影響を受けない，できる限り低レベルな入出力命令を用い，さらに，独自のデータバッファを設けるなどの手段が必要となる．しかし，このようなアプローチでは，OS が提供する機能の制限によって，DBMS の性能向上にも限界がある．例えば，入出力システムから直接ユーザのメモリ空間にデータ転送が行えなければ，メモリ間のデータ転送を増加させることになる．また，アプリケーションプログラムからページフィックスの指定を行えない場合には，DBMS の内部バッファ自体が OS のページングの対象となることも考えられる．

一方，近年，LAN の発達とともに LAN を用いて結合した分散データベースに対する要求が高まっている[12),13)]．この要求は，データの共有などの比較的単純なものから，動的負荷分散を考慮したトランザクションや演算のスケジューリングといった，より高度なものへと移っている．しかし，多くの OS が備えているネットワーク通信，分散ファイルシステム，遠隔手続き呼び出しなどの機能だけで，この要求に応えるのは困難である[14)]．分散 DBMS は，プロセスマイグレーションやネットワーク透過な記憶管理などの機能を OS に求めている．

以上のように，分散データベースにおいて性能の向上と，より高度な機能の実現を可能とするためには，分散データベース処理向きに OS を設計開発すべきである．我々は，この考えに基づいてデータベース OS μOPT-R を開発しており，その記憶管理およびデータ管理方式として，分散セグメンテーションを実現した．

2.2 μOPT-R の構成

(1) 対象システム

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，通信装置および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．

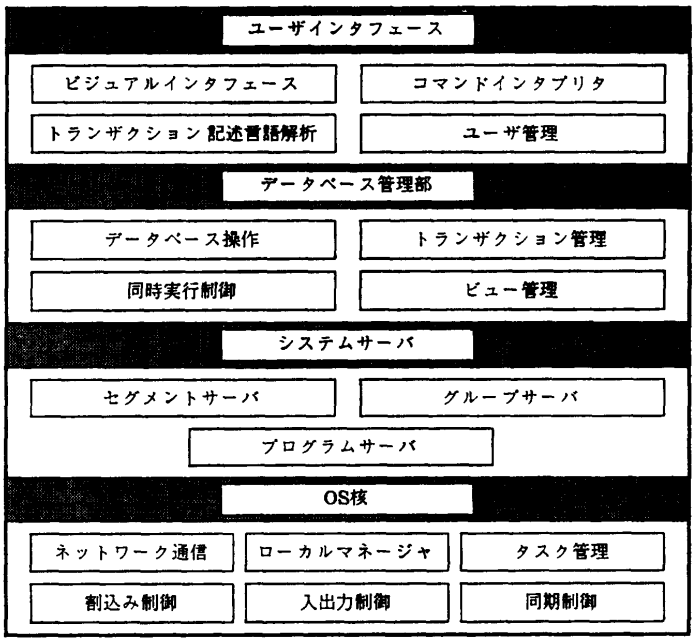


図1 μOPT-R のソフトウェア構成
Fig. 1 Software configuration of μOPT-R.

μOPT-R の現在の実験システムは，サイトに PC-9801 VM (NEC 製), LAN に BRANCH 4800 (NEC 製) を用いている．また，システムの記述言語はC言語と一部アセンブリ言語を使用している．

(2) ソフトウェア構成

μOPT-R のソフトウェア構成を図1に示す．各サイトには，それぞれ図1のモジュール群が配置される．また，モジュール群は，ユーザインタフェース，データベース管理部，システムサーバ，OS 核の4層で構成されている．

3. セグメントの構成

3.1 セグメント種別と論理的階層

μOPT-R の対象とするデータベースは，リレーショナル型のデータベースであり，データの論理構造はリレーション，すなわち“関係”である．“関係”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よって構成される．これらのセグメントは，論理的には木構造を成しており，一つの木は一つの“関係”を表現する．また，データベース全体では林構造となる．図2

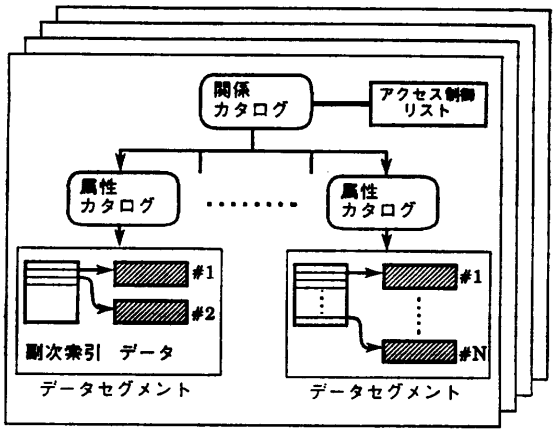


図 2 セグメントの論理的階層
Fig. 2 Logical hierarchy of segments.

にセグメントの論理的階層の一部を示す.

カタログセグメントはセグメント階層の上位に位置し，関係，属性に関する情報を格納するセグメントである．それぞれ，関係カタログ，属性カタログと呼ぶ．これらのセグメントは，メタデータであるデータディクショナリの役割を果たす．関係カタログには，“関係”に関する詳細な情報や，“関係”を構成する“属性”の一覧が格納される．属性カタログには，“属性”の型や属性値のとりうる最大値，最小値などの情報が格納される．

データセグメントは，属性値を格納するセグメントである．μOPT-R では，データセグメントの構造として階層転置型ファイル(HTF : Hierarchical Transposed File) を用いている[15]．HTF は，転置型ファイルに副次索引を付けたものである．

μOPT-R では，これらのセグメントに関する分散配置，アクセス制御，同時実行制御の各処理を，“関係”を単位として行っている．これとは別に，セグメントや“属性”などを処理単位とする方式も考えられる．しかし，その場合には，各処理が複雑になり，そのオーバヘッドが問題となる．また，ネットワークを介した通信量も増加する．μOPT-R では，処理速度の向上，特に通信量の削減に重点を置くこととし，“関係”を分散配置の単位としている．

また，各々の“関係”には，要求種別とその実行を許可するユーザ識別子を記述したアクセス制御リストを持たせることによってアクセス制御を行っている．“関係”の所有者は，各要求に対する実行可能ユーザリストを，設定/変更することができる．

3.2 セグメント識別子

分散システム上で，ネットワーク透過なセグメントアクセスを実現するには，分散システム全体で一意に識別可能なセグメント識別子が必要となる．識別子の実現には，識別子と物理アドレスが完全に独立な方式と，識別子中に何らかの位置情報を含む方式が考えられる．μOPT-R では，表1に示すように前者の方式を採用している．

データベースでは，ある“関係”がどのような“属性”を持つかといったメタデータ自体もデータである．すなわち，セグメント間の相互関係を示すデータ自身がデータベースの処理対象となる．また，データベースのデータは，システムの物理的な構成に依存すべきではない．したがって，μOPT-R におけるセグメント識別子は，物理アドレスを含まないものとした．**表1**に示すセグメント識別子が，そのまま，データベースにおけるデータ識別子となる．

3.3 セグメントの状態と遷移

セグメントは，トランザクションからの要求によって以下の6種類の状態を遷移する（**図3**参照）.

(1) 解 放 状 態

生成されたセグメントが，現在どのトランザクショ

表 1 セグメント識別子
Table 1 Segment identifiers.

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

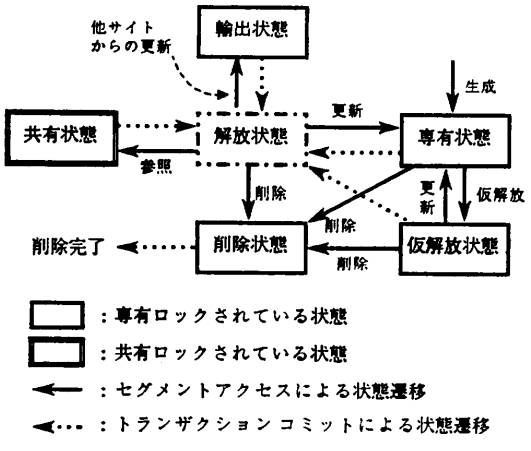


図 3 セグメントの状態遷移
Fig. 3 State transitions of segments.

ンからも使用されていない状態である．当該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，主記憶上に固定されている状態である．セグメントの更新要求によって，この状態へ遷移する．

(3) 共有状態

セグメントの参照要求の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．この状態では，セグメントの複製が，すべての要求元サイト上に作られるため，実体が複数存在することになる．

(4) 仮解放状態

他のセグメント要求を要因として，別のセグメントに置き換えら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．トランザクションのコミットにともなって，2次記憶上の実体も削除され，削除が完了する．

(6) 輸出状態

ローカルサイトの2次記憶にあるセグメントを，リモートサイトからの要求によって当該サイトの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コミット/終了時には，輸出先サイトのセグメントの内容が，輸出元サイトの2次記憶上に反映される．

4. セグメントアクセス方式

4.1 セグメントアクセスプロトコル

分散セグメンテーションは，OSI 参照モデルにおけるプレゼンテーション層に相当するレベルのサービスを提供している（**表2**参照）．表2において，同時実行制御，アクセス制御およびセグメント操作は，ユーザに対するサービスであり SVC (Supervisor Call) として提供される．また，トランザクション管理およびセグメント制御は，上記の三つの機能を実現するために用いられる．各サービスの概要を以下に示す．

表 2 セグメントアクセスサービス一覧
Table 2 List of segment access services.

サービス要素	オペレーション	機能概要
同時実行制御	LOCK	2相ロックのロッキングフェーズおよびトランザクションの開始
	UNLOCK	2相ロックのシュリンキングフェーズおよびトランザクションの終了
アクセス制御	SET	アクセス制御リストの設定
	CHANGE	アクセス制御リストの変更
セグメント操作	CREATE	セグメントの生成
	REFER	セグメントの参照
	UPDATE	セグメントの更新
	EXPAND	セグメントの拡張
	DELETE	セグメントの削除
	PRE_REL	セグメントの仮解放
トランザクション管理	WAKEUP	他トランザクションの終了待ちトランザクションの再開
	PREPARE	2相コミットの第1フェーズ
	COMMIT	2相コミットの第2フェーズ
	ROLLBACK	トランザクションの後退復帰
セグメント制御	LOCATION	セグメントの存在するサイト ID を得る
	EXPORT	主記憶上でのセグメント領域の解放およびセグメントの輸出状態への遷移
	IMPORT	セグメント輸入のための各種領域の確保

(1) 同時実行制御サービス

同時実行制御のためのユーザインタフェースである．μOPT-R では，同時実行制御方式として2相ロック方式を用いている．用意されているサービスは，2相ロック方式の第1相の要求を行う LOCK，第2相の要求を行う UNLOCK である．また，これらの要求は，トランザクションの開始および終了要求も兼ねている．各サービスは，SVC: lock() および unlock() としてユーザに提供される．

(2) アクセス制御サービス

アクセス制御リストの操作に関するユーザインタフェースである．ユーザは，所有しているセグメントのアクセス制御リストのみを操作可能である．

(3) セグメント操作サービス

ユーザが，セグメントを要求するためのインタフェースである．用意されている要求は，セグメントの生成，参照，更新，拡張，削除，仮解放の6種類で

```
lock (トランザクション ID, 関係 #1, 専有, 関係 #2,
      共有, …);
                ⋮
addressA=getseg (関係 #1, 属性A, …, 更新);
addressB=getseg (関係 #2, 属性B, …, 参照);
addressC=getseg (関係 #1, 属性C, …, 拡張);
                ⋮
  /* 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にアクセス */
                ⋮
unlock (トランザクション ID);
```

図 4 トランザクションにおけるアクセス手順
Fig. 4 Access sequence in a transaction.

ある．各要求は，セグメントアクセス用 SVC

char *getseg (segment_ID, operation_type);

の引数として指定される．生成，参照，更新，拡張の各要求に対しては，主記憶上にロードされたセグメントの先頭アドレスがユーザに対して返される．削除，仮解放の各要求は，セグメントの状態を変化させるためのものであり，その可否がユーザに対して返される．各要求に対する処理に関しては，次章で詳しく述べる．

(4) トランザクション管理サービス

この層は，トランザクションの再開や中断など，トランザクション管理に関するサービスを提供する．WAKEUP は，同時実行制御のロックに関係した処理を行うものである．また，PREPARE, COMMIT, ROLLBACK は，トランザクションの 2 相コミットに関する処理を行う．

(5) セグメント制御サービス

この層では，セグメントの状態に関する制御を行う．LOCATION は，セグメントの存在するサイトを特定するサービスである．EXPORT と IMPORT は，セグメントの輸出入に伴う処理を行う．

4.2 セグメントのアクセス手順

トランザクションにおけるセグメントへのアクセス手順を図 4 に示す．トランザクションでは，まずSVC: lock() によって，使用するすべてのセグメントをロックしなければならない．すべてのセグメントを一度にロックするのは，デッドロックを避けるためである．その後，必要に応じて SVC: getseg() により各セグメント要求を発行し，セグメントの先頭アドレスを得る．トランザクション内では，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にアクセスする．セグメントへのアクセスをすべて終了した後，トランザクションは SVC: unlock() を発行する．これによって，セグメントの解放およびトランザクションのコミット処理を行う．セグメントは，getseg() (仮解放要求を除く) が発行されてから，unlock() までの間，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．

5. セグメント管理部の構成と処理方式

5.1 構　　成

μOPT-R のセグメント管理部の構成を図 5 に示す．セグメント管理部は，ローカルマネージャ，セグメントサーバ，同時実行制御部から構成されている．以下，これらについて順に説明する．

(1) ローカルマネージャ (LM)

LM の主な機能は，主記憶/2 次記憶間のセグメントの置き換え操作である．主記憶，2 次記憶およびセグメントはページ (1024 バイト) を単位として管理されている．現在，μOPT-R の開発マシンは，MMU (Memory Management Unit) を備えていないため，ページングを用いた仮想記憶方式は実現していない．また，セグメンテーションにおけるセグメントの置き換え，すなわち主記憶への読み込みおよび 2 次記憶への追い出しに関する処理は，すべてソフトウェアで行っている．

セグメントの読み込みは，ユーザが発行したセグメ

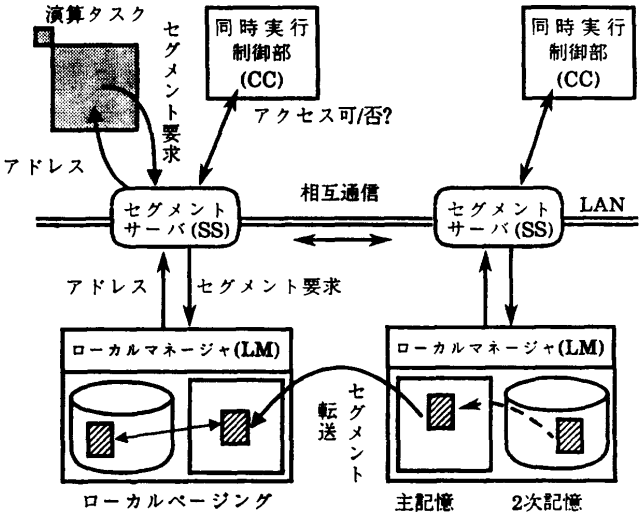


図 5 セグメント管理部の構成
Fig. 5 Configuration of the segment manager.

ントの参照/更新/拡張要求に対してオンデマンドに行われる．また，主記憶領域の割り付けは，以下のFINUFO (First In Not Used First Out) 方式[16]で行う．

- 主記憶を下位アドレスから上位アドレスへ走査し，未使用領域の中から First-Fit で割り付ける．
- 次回の割り付け時には，今回割り付けた領域の次の領域から走査する．
- 最上位アドレスへ到達したら，最下位アドレスから走査開始点まで走査する．

さらに，未使用領域の割り付けができずにセグメントの置き換えを行う場合に備えて，この走査と同時に，次の各状態にある領域を洗い出しておく．

- 状態1：未変更のセグメントが存在するページ
- 状態2：変更されたセグメントが存在するページ

セグメントの置き換えが必要となった場合には，まず状態1そして状態2の順で，対象領域を順次拡大して，割り当て可能領域を決定する．また，割り当て領域に状態2の部分が含まれる場合には，2次記憶へ追い出すセグメント数が最小になるように選択する．

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．しかし，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[15]，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．したがって，外部フラグメンテーションは問題とはならない．

(2) セグメントサーバ (SS)

ユーザが SVC を用いて行ったセグメント要求は，OS 核を介して，SS へ中継される．SS は，各サイトの SS 間で相互にセグメント情報およびセグメント本体を送信することによって，リモートサイト上のセグメントに対する要求を処理する．また，SS は，システム全体で，矛盾なくセグメント識別子が用いられるように，識別子からのセグメント位置の特定や，識別子作成に関する管理を行う．

(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み込むことによって，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．これに関しては，6章で詳述する．

5.2 処理の概要

トランザクションは，トランザクション管理部によって解析され，複数の関係演算に分解される．各関係演算は，それぞれ一つのタスク（これを演算タスクと呼ぶ）として実現されており，演算タスクが，セグメント要求やタスク間通信を行うことによって，トランザクションの処理を行う．演算タスクにおいて発行されたセグメント要求は，セグメント管理部の内部では以下の手順で処理される．

(1) セグメント要求は 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 に対してブロードキャストし，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 SS_R，要求元サイトの SS を SS_L とすると，SS_R は，(4a)～(5a) の手順でセグメントの先頭アドレスを得る．

(6b) SS_R は，SS_L に対してセグメントの内容を転送する．

(7b) SS_L は主記憶上に領域を確保し，SS_R 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) SS_L は，領域の先頭アドレスを演算タスクへ返す．

5.3 各セグメント要求に対する処理

セグメント管理部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．

(1) 参照/更新/拡張要求に対する処理

参照の場合，一つのセグメントの複製が複数のサイト上に存在する（各サイトには高々一つ）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，一つの実体のみが主記憶上に存在する．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，複数のサイトから一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の異なる演算タスクからの要求であっても許されない．

（2） 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2次記憶上に実体を残したまま削除状態に遷移させる．トランザクションがコミットされた時点で，2次記憶上の実体も削除され，削除が完了する．

（3） 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上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．

生成の対象が，関係カタログの時には，要求元サイト上に，新たな“関係”が生成される．この場合，ロッキングに関して問題が生じる．ここで生じる問題とは，生成要求が行われるまで，“関係”がロックされないため，複数のトランザクションが，同時に同じ識別子を持つ“関係”を生成してしまう危険があるということである．さらに，μOPT-R では，識別子中にサイト識別子が含まれていないため，この問題は異なるサイトにまたがって生じる可能性がある．そこで，“関係”の生成時には，まず，ローカルサイト上で“関係”を生成した後，すべてのサイトへ問い合わせを行い，既に同じ“関係”が存在する場合には，不当要求ということでトランザクションを中断させる．同じ“関係”が存在しない場合には，生成した“関係”を専有ロック状態へ遷移させる．

また，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生成したものとする．

（4） 仮解放要求に対する処理

演算タスクは，いったんセグメント要求 SVC: getseg() によってセグメントの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置き換え不可能である（μOPT-R では，セグメント置き換えに関する全処理をソフトウェアで実現しているため）．しかし，トランザクション内で，ある一定期間使用しないセグメントを，主記憶上に放置するのは効率が悪い．そこで，それ以降，再度セグメント要求を発行するまではポインタによるアクセスを行わないという条件の下で，トランザクション内部でも，使用中のセグメントを置き換え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，セグメント管理部に対して再度セグメント要求 SVC: getseg() を発行しなければならない．演算タスクにおいては，セグメントへのアクセスを行わないことを予測可能な区間で仮解放要求を行うことによって，そのセグメントの主記憶領域を解放し，主記憶を有効に利用することができる．

μOPT-R では，主記憶と2次記憶間でのセグメントの内容の同期は，トランザクションのコミットを契機として行われる．すなわち，主記憶上のセグメントの更新は，トランザクション終了まで2次記憶に反映させる必要がない．そこで，リモートサイトから輸入しているセグメントが，トランザクション実行中に仮解放状態となり，ページアウトの対象となった場合には，それを書き出す2次記憶として，輸出元サイトではなく，ローカルサイトの2次記憶を利用している．

6. 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．我々の方式は，Gray による2相ロック方式[17]を，デッドロックフリーで行えるように変更したものである．本章では，この方式に基づくμOPT-R の同時実行制御について述べる．

6.1 トランザクションの開始処理

μOPT-R では，すでに述べたように“関係”を単位としたロッキングを行っている．また，ロッキング種別としては，一つのトランザクションがセグメントを専有して更新/拡張/削除処理を行う場合の専有ロックと，セグメントの参照のみを行い，複数のトランザクションがセグメントを共有することが可能な場合の共有ロックを設けている．

2相ロック方式ではデッドロックの発生が問題となる．μOPT-R ではロッキングフェーズの処理を論理的に不可分なものとし，他のトランザクションにおけるロッキングフェーズと排他的に実行することによってこれを回避している．ロッキングフェーズの処理手順を以下に示す．

(1) トランザクションは SVC: lock() によって，ロックしたい“関係”名（実際には関係カタログの識別子）の列を同時実行制御部 (CC) に要求する．

(2) CC は，“関係”の存在する各サイトの CC と通信して，“関係”を一つずつ仮ロックしていく．この時，必ず“関係”名を辞書順に並べた順で要求を行う．

(3) 一つでも，仮ロックに失敗したらその時点で，すべての仮ロックを白紙に戻し，トランザクションを待ち状態にする．

(4) すべての仮ロックに成功した場合には，仮ロックを真のロックへ移行して，ロックを完了する．

μOPT-R では，仮ロックの要求を“関係”名の辞書順で行うことによって，トランザクションの相互封鎖 (mutual blocking) を防いでいる．

6.2 トランザクションのコミット

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，輸出元および輸出先でそれぞれ次の処理が行われる．輸出元では，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出元サイトへ送信し，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．

7. 性能評価

本章では，μOPT-R において提供されている各セグメント要求に関して，処理時間を測定した結果を示す．実験は，2章で述べた実験システム（サイト数4）で，特定の2サイト間で行った．ここでの処理時間とは，演算タスクからセグメント要求を発行し，結果が演算タスクに返されるまでの時間である．また，対象となるセグメントには関係カタログ（1024 バイト）を用いた．

表3に，セグメントがローカルサイトおよびリモートサイトに存在する場合の，各セグメント要求の処理に要する時間を示す．表3の主記憶上および2次記憶

表 3 セグメント要求の処理時間（単位はミリ秒）
Table 3 Execution times for segment requests (Computed times are in msec.).

当該セグメントの位置	ローカルサイト				リモートサイト			
	主記憶上（読み込みなし）		2次記憶上（読み込みあり）		主記憶上（読み込みなし）		2次記憶上（読み込みあり）	
他セグメントの追い出し	なし	あり	なし	あり	なし	あり	なし	あり
生成	19.0	192.3	—	—	892.8 (1/0)	1084.5 (1/0)	—	—
更新	13.2	—	188.3	366.1	3088.4 (1/N)	3130.2 (1/N)	3274.3 (1/N)	3317.2 (1/N)
参照	12.9	—	184.7	362.8	3028.7 (1/N)	3062.4 (1/N)	3212.7 (1/N)	3251.1 (1/N)
削除	14.2	—	—	—	894.3 (1/0)	—	—	—
仮解放	11.5	—	—	—	895.2 (1/0)	—	—	—

下段括弧内はセグメントのページ数を N とした場合の通信回数を示す（制御情報/セグメント本体）．
—：要求不可能，N：セグメントのページ数．

上とは，各々要求されたセグメントが主記憶上に存在する場合と2次記憶上に存在する場合を表す．さらに，セグメント置き換えの有無は，当該セグメントを主記憶に読み込むための空き領域を得るために，他のセグメントを2次記憶に追い出すか否かを示している．以下では，特に更新要求に関して考察する．

(1) セグメントがローカルサイト上に存在する場合

表3から，セグメントが主記憶上に存在する場合は，13.2ミリ秒で更新要求が処理されることが分かる．また，セグメント置き換えが発生する場合には，1回の入力または出力につき処理時間が170から180ミリ秒増加していることが分かる．ここで，SSを用いずに直接LMに更新要求を行った場合の処理時間(入出力が行われない場合)は，3.2ミリ秒であることが分かっている．これと，表3のローカルの場合とを比較することによって，ネットワーク通信を除いた，SSにおける処理のオーバヘッドが，10ミリ秒(=13.2−3.2)であることが分かる．この値は，セグメント置き換えのための入出力と比較してほとんど無視できる値であると考えられる．

(2) セグメントがリモートサイト上に存在する場合

各要求に応じて表3の括弧中で示される回数分のネットワーク通信が行われる．ネットワーク通信としては，セグメント要求パラメータを含む制御情報(約40バイト)の送信と，セグメント本体の転送の2種類がある．制御情報の送信は，セグメントのサイズの要求種別にかかわらず，各要求につき1回行われる．この制御情報の通信には，μOPT-Rの一斉送信機能を用いている．μOPT-Rの一斉送信では，多重アクセスバス型のLANに対するハードウェアブロードキャストを用いており，サイト数に影響されずほぼ一定の時間での通信が可能である．また，セグメント本体の転送は，セグメントのページ(1024バイト)を単位として行う．したがって，セグメントのページ数をNとすると，1回の要求につきN回の通信が行われる．現在の実験システムでは，制御情報(約40バイト)の一斉送信に875.0ミリ秒，1回のセグメントの転送(1024バイト)に2145.2ミリ秒を必要とする．

さらに，リモートサイト上でセグメントが2次記憶上に存在する場合には，セグメントの読み込みを行う．また，それとは別に，ローカルサイト上で主記憶に空き領域がない場合には，セグメントの追い出しを行う．以下，各場合について考察する．

(a) 主記憶上に存在する場合

セグメントがリモートサイトの主記憶上に存在し，しかも，ローカルサイトの主記憶に空き領域がある場合には，セグメントの置き換えは発生しない．この場合，表3の更新要求に対する処理時間を見ると，その値は，3088.4ミリ秒である．このうち，ネットワーク通信に要する時間が，3020.2ミリ秒(=875.0+2145.2)であるので，通信を除いた処理のオーバヘッドは，68.2ミリ秒(=3088.4−3020.2)である．この値は，ローカルサイト上のセグメントに対する更新要求の処理時間，13.2ミリ秒より，若干大きな値となっているが，これは，リモートサイトで，セグメントを輸出状態とするための処理が加わるためである．

また，ローカルサイトの主記憶に空き領域がない場合には，セグメントの追い出しが行われる．その場合には，追い出しを行わない場合と比較して，表3より41.8ミリ秒(=3130.2−3088.4)しか処理時間が増加していない．これは，μOPT-Rのマルチタスク機能により，通信と入出力がオーバラップして処理されたためである．

(b) 2次記憶上に存在する場合

この場合，(a)の処理にセグメントの読み込み処理が加わる．処理時間は，セグメント追い出しが発生しない場合で3274.3ミリ秒，発生する場合で3317.2ミリ秒である．これらの値は，(a)の場合と比較して，185.9あるいは187.0ミリ秒増加している．この増加分は，セグメント読み込みにおける入出力時間であると考えられる．

すでに述べたように，ローカルサイトの主記憶上に空き領域がない場合には，セグメント追い出しの処理が加わるが，(a)の場合と同様に処理時間は，入出力が1回増えたにも関わらず，約42.9ミリ秒(=3317.2−3274.3)しか増加していない．これも，通信と入出力がオーバラップして処理されたためである．ただし，セグメント読み込みの場合には，同一のセグメントに対して読み込みと送信が行われるため，この効果は得られていない．

以上考察してきたように，セグメント管理部における処理時間の大部分は通信のための時間であり，それ以外のオーバヘッドは問題にならないと考えられる．したがって，高速な通信媒体を使用することによって，その処理時間が大幅に改善されることが期待できる．これに関しては，今後の課題である．

8. おわりに

本論文では，μOPT-R における分散セグメンテーションの概要について述べた．セグメント管理部はすでに完成しており，分散セグメンテーションの実現可能性を確認することができた．さらに，本論文では，実験システム上での性能測定を行った．測定結果は，分散セグメンテーションの有効性を十分に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，より高速な通信媒体や，TCP/IP プロトコルのサポートなど，今後の改善が必要であると考えている．また，本実験システムは，セグメンテーションやページングをサポートするハードウェアを備えていないため，メモリ保護が十分に行えないという問題がある．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．

謝辞 日頃から，貴重な御意見，御討論を頂いている國枝義敏助手，岡部寿男助手を始めとする津田研究室の諸氏に感謝いたします．

なお，本研究は一部文部省科学研究費（課題番号 01580025）による．

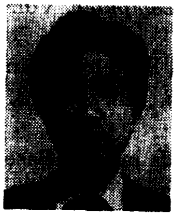
参考文献

1) Stonebraker, M.: Retrospection on a Database System, *ACM TODS*, Vol. 5, No. 2, pp. 225-240 (1980).
2) Stonebraker, M.: Operating System Support for Database Management, *Comm. ACM*, Vol. 24, No. 7, pp. 412-418 (1981).
3) Spooner, D., Gudes, E. and Fischer, P.: The Logical Object Model for Data Base Operating Systems, *COMPSAC*, pp. 769-775 (1980).
4) Hsiao, D. K.: A Study of the Relationship of Database and Operating Systems, *Proc. 3rd International Conf. on Data Engineering*, pp. 22-27 (1987).
5) 國枝和雄，水谷高康，大久保英嗣，津田孝夫：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 回情報処理学会全国大会論文集，5P-9 (1988).
6) 國枝和雄，水谷高康，大久保英嗣，津田孝夫：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，情報処理学会データベースシステム・オペレーティングシステム合同研究会資料，DBS-73-5/OS-44-5 (1989).
7) 水谷高康，國枝和雄，大久保英嗣，津田孝夫：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 回情報処理学会全国大会論文集，5P-8 (1988).
8) 水谷高康，國枝和雄，大久保英嗣，津田孝夫：データベース専用オペレーティングシステム μOPT-R における演算処理の高速化，情報処理学会データベースシステム・オペレーティングシステム合同研究会資料，DBS-73-6/OS-44-6 (1989).
9) Bach, M. J.: *The Design of the UNIX Operating System*, Prentice-Hall (1986).
10) Rosell, J. R.: Empirical Data Reference Behavior in Data Base Systems, *Computer*, Vol. 9, No. 11, pp. 9-13 (1976).
11) Smith, A. J.: Sequentiality and Prefetching in Database Systems, *ACM TODS*, Vol. 3, No. 3, pp. 223-247 (1978).
12) Stonebraker, M.: Future Trends in Data Base Systems, *Proc. 4th International Conf. on Data Engineering*, pp. 222-231 (1988).
13) 日経コンピュータ別冊：市場形成に向けて製品化始まる分散データベース，pp. 145-159，日経マグロウヒル社 (1988).
14) Tanenbaum, A. S. and Mullender, S. J.: Operating System Requirements for Distributed Data Base Systems, in Schneider, H. J. (ed.), *Distributed Data Bases*, pp. 105-114, North-Holland Publishing Company (1982).
15) 大久保英嗣，津田孝夫：階層配置型ファイルに基づく関係操作アルゴリズム，情報処理学会論文誌，Vol. 26, No. 1, pp. 130-138 (1985).
16) Corbato, F. J.: A Paging Experiment with the MULTICS System, *Report MAC-M-384, Project MAC*, MIT Press (1968).
17) Eswaran, K. P., Gray, J. N., Lorie, R. A. and Traiger, I. L.: The Notion of Consistency and Predicate Locks in a Database System, *CACM*, Vol. 19, No. 11, pp. 624-633 (1976).

(平成2年7月 9 日受付)
(平成3年1月11日採録)

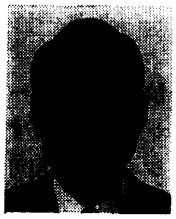
國枝　和雄（正会員）

昭和39年生．昭和62年京都大学工学部情報工学科卒業．平成元年同大学院修士課程修了．現在，同大学院博士課程在学中．オペレーティングシステム，データベースシステム，分散処理等に興味を持つ．日本ソフトウェア科学会会員．



大久保英嗣（正会員）

昭和26年生．昭和49年北海道大学理学部数学科卒業．昭和52年同大学工学部情報工学科大学院修士課程修了．同年(株)日立製作所ソフトウェア工場に入所．主としてFORTRANコンパイラの開発に従事．昭和54年より京都大学工学部情報工学科助手．昭和60年同講師，昭和62年同助教授，現在に至る．工学博士．オペレーティングシステム，データベースシステム等の研究に従事．日本ソフトウェア科学会，日本ロボット学会各会員．



津田　孝夫（正会員）

1957年3月，京都大学工学部電気工学科卒業．現在京都大学工学部情報工学科教授．計算機ソフトウェア講座担当．工学博士．自動ベクトル化/並列化コンパイラ，スーパコンピューティング，オペレーティングシステムの研究に従事．「モンテカルロ法とシミュレーション」(培風館)，「現代オペレーティングシステムの基礎」(オーム社，共著)，「数値処理プログラミング」(岩波書店)などの著書がある．昭和63年度情報処理学会論文賞受賞．本学会関西支部長．ACM，SIAM 各会員．IFIP/WG 2.5 (Numerical Software) 委員．