データベースシステム 116－9
（1998. 7. 8）

並列論理型言語によるDBMSのソフトウェアオーバヘッドの削減

宮崎 純
miyazaki@jaist.ac.jp
北陸先端科学技術大学院大学
情報科学研究科
〒923-1292 石川県能美郡辰口町旭台 1-1

横田 治夫
yokota@cs.titech.ac.jp
東京工業大学大学院
情報理工学研究科 計算工学専攻
〒152-8552 東京都目黒区大岡山 2-12-1

あらまし 我々は，並列論理型言語 KLIC を用いて，並列アクティブデータベースのプロトタイプ Parade (PARallel Active Database Engine) を実装してきた．しかしながら，KLIC の標準データ型のみでのナイーブな実装では，I/O オーバヘッド，ごみ集め，関係代数演算のオーバヘッドが大きい．そこで，KLIC のジェネリックオブジェクトを用いて，新しいデータ型であるページオブジェクトを定義し，これらのオーバヘッドを削減した．本稿では，ページオブジェクトによる実装での性能評価を行い，並列記述が容易な高級言語による低オーバヘッドの並列 DBMS の可能性を示す．

和文キーワード データベース，並列論理型言語，ジェネリックオブジェクト，性能評価

Improvement of Software Overhead in a DBMS Implemented by a Parallel Logic Programming Language

Jun MIYAZAKI
miyazaki@jaist.ac.jp
School of Information Science,
Japan Advanced Institute of
Science and Technology
1-1 Asahidai, Tatsunokuchi,
Ishikawa 923-1292, Japan

Haruo YOKOTA
yokota@cs.titech.ac.jp
Graduate School of
Information Science and Engineering,
Tokyo Institute of Technology
2-12-1 Ookayama, Meguro,
Tokyo 152-8552, Japan

Abstract We have implemented a parallel relational active database system, named Parade, using a parallel logic programming language, KLIC. However, the naive implementation using standard data types in KLIC causes a grate deal of overheads in accessing I/O, garbage collection, and relational algebra operations. Toward the high performance system, we defined a new data type, called page object, utilizing a generic object function in KLIC. We reimplemented the system using the page object to decrease the overheads. In this paper, we evaluate the performance of the new system, and show a possibility of a low overhead DBMS implemented by a high level parallel programming language.

英文 key words
database, parallel logic programming language, generic object, performance evaluation

1 はじめに

近年，高度情報化社会の進展とともに，多様かつ大量のデータがデータベース上に蓄積され，多くの人々がそのデータを利用している．扱うデータ量の増大，およびデータマイニングなどデータベースの利用方法の多様化に伴い，並列処理によるデータベースの性能向上が望まれている．

その中で，最近注目されているアクティブデータベース [1] は，アクティブルールによりデータベース処理を能動的に起動することのできるシステムである．アクティブデータベースは，ルールにより自動的にデータベース操作が行われるため，従来のパッシブなデータベースよりも負荷が高く，高性能な DBMS が要求される．データベース処理には多くの I/O やメモリ参照が行われるため，並列化のためには無共有並列計算機が適する．

我々は，並列論理型言語 KLIC[2] を用いて，並列アクティブデータベースシステムのプロトタイプ Parade (PARallel Acitive Database Engine)[3, 4, 5] を実装してきた．KLIC の論理変数によるプロセス間のメッセージ通信は，無共有並列計算機上でデータベース処理を実現する際に非常に適合性が良い．さらに，アクティブルールのトリガ処理に KLIC のプロセスの suspend/resume 機能を利用すると，記述が容易である．

これまでの研究により，データベース処理を KLIC のプロセス間のメッセージ通信を利用することにより実現し，汎用ワークステーションならびに超並列計算機 nCUBE2 上で動作させ，その記述能力と移植性の高さを実証してきた．しかしながら，KLIC の標準で用意されているデータ型のみでのナイーブな実装では，(1) 二次記憶および PE 間のデータ転送などの I/O 処理のオーバヘッド，(2) ごみ集めのオーバヘッド，(3) プロセススイッチのオーバヘッド等が非常に大きく，システムの性能のボトルネックとなっていることが分かってきた．これは，KLIC が I/O 性能やプロセスの粒度に重点を置いて設計されたものではないことに起因する．

KLIC にはデータ型を拡張可能なよう，ジェネリックオブジェクトと呼ぶデータ構造が用意されており，ユーザが任意にデータ構造を定義できる．また，そのデータ構造に対して自分自身の処理方法をメソッドとして定義できる [6]．このジェネリックオブジェクトを用いて，Parade の並列データベース処理が実用的な速度で動作可能なよう，新しいデータ構造であるページオブジェクトを定義し，DBMS の基本性能の向上や関係代数演算のメソッド化による性能向上を試みた．これまでの研究で，ページオブジェクトの導入により，基本 I/O 性能が改善できることが分かった [7]．本論文では，ページオブジェクトを既存の Parade の DBMS に導入し，その性能を測定した．

2 Parade

並列アクティブデータベースプロトタイプ Parade は，超並列計算機をターゲットとしたクライアント/サーバ型のアクティブデータベースである [4, 5]．Parade は並行細粒度プロセスを記述できる並列論理型言語 KL1 で記述され，KLIC[2] により C 言語へ変換することにより，超並列計算機 nCUBE2 や分散ワークステーション上で動作する．Parade は図 1 のように，関係データベースにアクティブルール処理機構を統合した形で構成される．各要素は KLIC の並行プロセス集合から構成され，個々のプロセスは自然な形で同期がとられ，また並列に動作する．

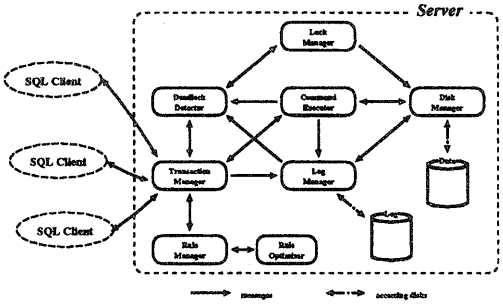

図 1: Parade のシステムアーキテクチャ

3 ページオブジェクト

KLIC は Lisp や Prolog などの言語と同様に，ごみ集め (GC) がシステムにより自動的に行われ，copying GC が実装されている [6]．また，データは KLIC のアトミックなデータ型か，そのリストとして扱われる．KLIC によるナイーブな DBMS

の実装では，タップルの各々の属性値がKLICのアトミックなデータとして扱われ，タップルはそれらのリストとして表現される(図2(a)参照)．このため，

(1) サイズの小さな属性値を単位としてI/Oが行われるため，I/Oオーバヘッドが大きい(図3参照)．

(2) 全てのデータはKLICのヒープ上に置かれ，GCの対象となる(図4(a)参照)．大量のデータを扱うデータベースの場合，GCのオーバヘッドは大きい．

(3) 関係代数演算はリストに対するデータ操作で実現されるため，リストを手繰るオーバヘッドやプロセススイッチのオーバヘッドが生じる．

などの問題点が生じる．

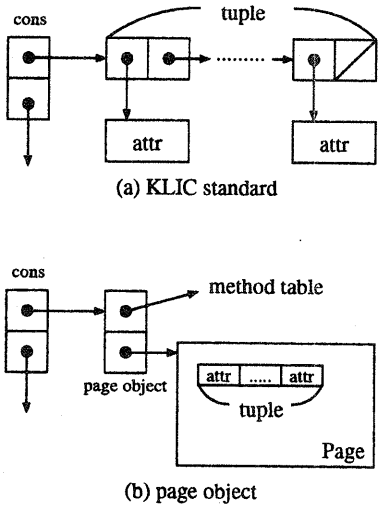

図 2: タップルのデータ構造

これらの問題点を解決するために，KLICのジェネリックオブジェクトを用いて，ページオブジェクトを定義した(図2(b)参照)．ページ本体の構成は，一般的な実装方式を用いた[8]．このページオブジェクトの導入により，I/Oオーバヘッドの削減，GCオーバヘッドの削減および関係代数演算の効率化が実現できる．

I/Oオーバヘッドの改善に関しては，ページオブジェクトによるページ単位のディスクアクセス

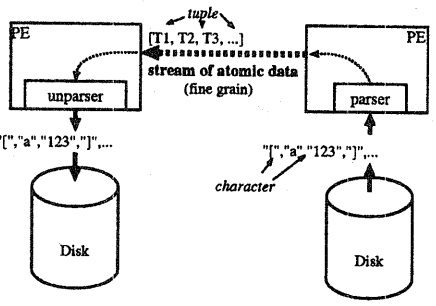

図 3: KLIC 標準のI/O

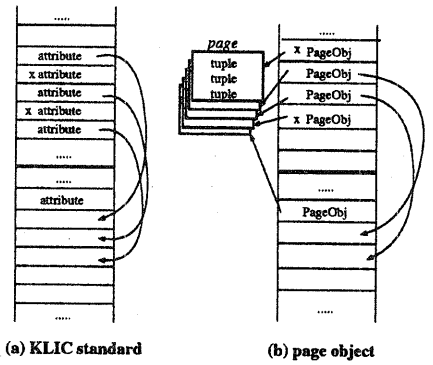

図 4: ごみ集め

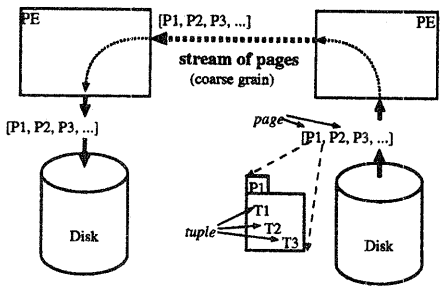

図 5: ページオブジェクトを用いたI/O

によりスループットが改善されるだけでなく，並列B-Tree[9]などのアクセスパス構造との親和性の向上にも貢献する．ページ単位でのPE間のデータ転送を行なうことにより，通信のスループットの向上も同時に達成することができる(図5参照)．

GCのオーバヘッドの改善に関しては，図2(b)のページ本体をKLICのヒープ外に配置することにより，GCの対象となるのはメソッドテーブルへのポインタとページ本体へのポインタを持つページオブジェクト本体のみとなる．これにより無駄なデータの移動が減少し，GCのオーバヘッドが改善される(図4(b)参照)．

関係代数演算の性能改善に関しては，関係代数演算をページオブジェクトに対するメソッドとして定義することにより，リストを手繰るオーバヘッドの削減や，ページ内のタップルに対して一括して演算を行えるため，プロセススイッチの回数は減少しそのオーバヘッドを軽減できる．

4 ページオブジェクトによるオーバヘッドの改善

ページサイズを4KBとしてページオブジェクトを定義し，このページオブジェクトを用いて，以下の各節の性能の測定を行った．データベースには，ウイスコンシンベンチマークに準拠するテーブル(1タップル208バイト)を利用した．

4.1 I/O性能の改善

ページオブジェクトとKLIC標準のI/Oの性能の比較するために，並列計算機nCUBE2 (20MHz)と，SunOS4.1.4が稼働しているSun SparcStation 5 (170MHz)を10Mbpsのレイヤー3 Ethernetスイッチで接続したWSクラスタを用いて行った．nCUBE2のI/O性能は，ディスクからの読み込み，書き込み，および通信スループットは，それぞれ1.1MB/s，0.46MB/s，2.2MB/sである．なお，用いたデータベースのサイズは3000タップルである．

測定結果を表1に示す．この結果からも分かるように，ページオブジェクトによるI/Oは，KLIC標準のI/Oに比較して著しいI/O性能の向上が得られることが明らかである．これは，ページオブジェクトによるI/Oが，大きなサイズのデータ単位でI/Oを行なうため，ディスクI/O，通信の両方のスループットが向上したからである．これに対してKLIC標準のI/Oは，その処理粒度が小さいため，I/Oには大きなオーバヘッドが伴う．

4.2 ごみ集めのオーバヘッドの改善

KLIC標準のデータ構造では，全てのデータはKLICのヒープ上に置かれる．KLICはcopying GCを用いており，GCが起きたとき，データはヒープ上の異なるアドレス空間へコピーされる．一方，ページオブジェクトを用いた場合，ページ本体はKLICのヒープ外に置くため，GCの対象となるのはメソッドへのポインタとページへのポインタを保持しているページオブジェクト本体と，ページオブジェクトを指すconsセルのみである．Copying GCのオーバヘッドは，データの移動のコストでありデータ量が多いほどそのコストは大きい．すなわち，ページオブジェクトにより，GCのオーバヘッドが削減される．

図6は，10000タップルのテーブルを用いて，GCを強制的に1000回起したときの処理時間を，SparcStation 5上で測定したものである．

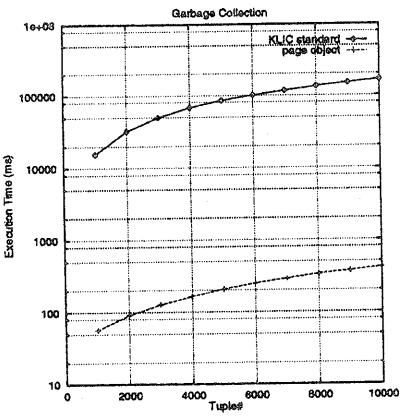

図6: ごみ集めのオーバヘッド

測定結果より，処理時間$E(\mu s)$とタップル数Tの関係は，KLIC標準の場合

$$E = 16.7T - 358$$

に対して，ページオブジェクトの場合は

$$E = 0.040T + 9.87$$

表 1: I/O 性能の比較 (単位: msec)

	nCUBE2		WS Cluster	
	KLIC starndard	page object	KLIC standard	page object
Disk Read	25.6×10^3	907	4630	76.5
Disk Write	235×10^3	2030	9590	102
PE 間通信	4170	165	36.0×10^3	1230

となった．つまり，ページオブジェクトによる実装は，GC のオーバヘッドを約 1/400 に改善したことになる．

ページオブジェクトの場合，1 ページのデータを保持するために，ページオブジェクトを指す cons セルが 1 つ (8 バイト) と，ページオブジェクト本体のデータ構造が 1 つ (8 バイト) が必要である．よって，GC の対象となるのは 1 ページ当り 16 バイトである．1 ページ 4KB であるから，1 ページに 19 タップル保存され，1 タップル当りの GC の対象となるデータは 0.842 バイトとなる．

一方，KLIC 標準のデータ構造の場合，1 タップル全体を指す cons セルが 1，タップル内では各属性を指す cons セルが 16 必要である．属性値のためのヒープ 208 バイトと合わせて，1 タップル当り合計 344 バイト必要である．これらは全て GC の対象となる．

以上の 2 方式を比較すれば，GC によるメモリ間のデータ転送コストは，KLIC 標準に対してページオブジェクトは約 1/400 になり，実測データに一致する．

4.3 関係代数演算性能の改善

ページオブジェクトに対して，選択，射影，結合演算をメソッドとして定義した．これらの演算のメソッド化により，関係代数演算の処理性能がナイーブな実装に対してどれだけ向上したかを測定した．

4.4 ページオブジェクトによる関係代数演算の設計

各ページオブジェクトに対するメソッド呼び出しが並行に動作可能なよう，図 7 のように各演算を実装した．入力ストリームから入力されたページオブジェクトは，ディスパッチャにより，各ページオブジェクトごとにプロセスを生成し，そのプロセス内で選択演算メソッド，射影演算メソッド，または結合演算メソッドが呼び出される．これらのプロセスは，異なる PE に割り当てることができ，並列に動作可能である．各メソッドからの出力ページオブジェクトは，マージャにより出力ストリームにまとめられる．

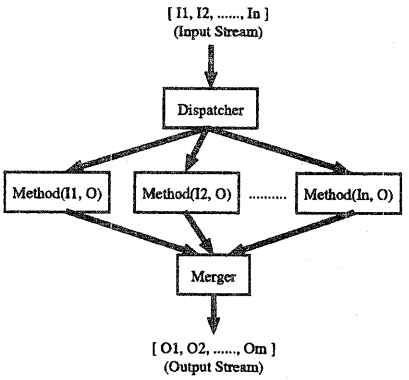

図 7: 関係代数演算の設計

4.4.1 選択演算

以下の選択演算を行う SQL 命令を発行した時の処理時間を図 8 に示す．なお選択演算では，全てのタップルが選択条件に適合する．

```
SELECT *
FROM   table
WHERE  table.unique1 >= 0
AND    table.unique1 <= 10000
```

測定結果から，処理時間 E(ms) とタップル数 T の関係は，KLIC 標準の場合

$$E = 0.539T - 53.5$$

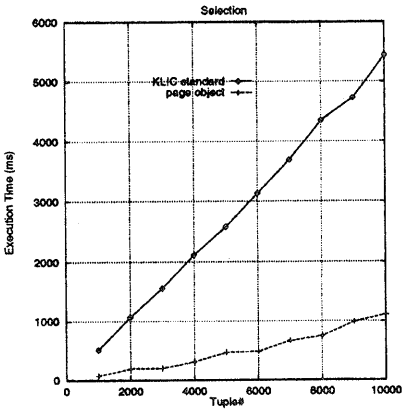

図 8: 選択演算の性能

に対して，ページオブジェクトの場合

$$E = 0.113T - 97.9$$

となった．選択演算のページオブジェクトに対するメソッド化により，ナイーブな実装と比較して約 4.8 倍の処理速度の向上が達成された．

4.4.2 射影演算

以下の射影演算を行う SQL 命令を発行した時の処理時間を図 9に示す．

```
SELECT table.unique1
FROM   table
```

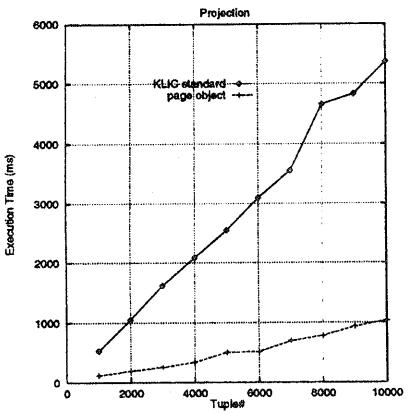

図 9: 射影演算の性能

測定結果から，処理時間 E(ms) とタップル数 T の関係は，KLIC 標準の場合

$$E = 0.546T - 70.8$$

に対して，ページオブジェクトの場合

$$E = 0.104T - 35.9$$

となった．射影演算のページオブジェクトに対するメソッド化により，ナイーブな実装と比較して約 5.3 倍の処理速度の向上が達成された．

4.4.3 結合演算

結合演算は，ナイーブな実装ではネステッドループ方式のみが実装されている．比較のため，ページオブジェクトによる実装もネステッドループ方式を実装し，両者の比較を行った．比較のために，以下の結合演算を行う SQL 命令を発行したときの処理時間の測定を行った．結果は図 10に示す．

```
SELECT a.*, b.*
FROM   table AS a, table AS b
WHERE  a.unique1 = b.unique1
```

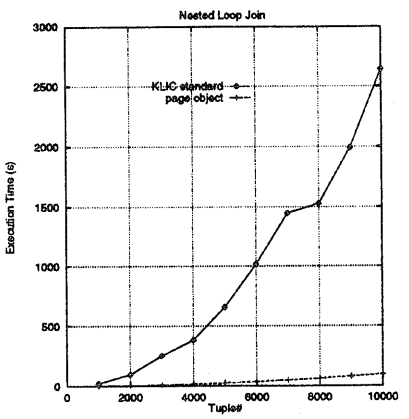

図 10: 結合演算の性能

測定結果から，処理時間 E(s) とタップル数 T の関係は，KLIC 標準の場合

$$E = 23.5 \times 10^{-6}T^2 + 24.8 \times 10^{-3}T - 38.5$$

に対して，ページオブジェクトの場合

$$E = 0.965 \times 10^{-6}T^2 + 11.1 \times 10^{-6}T - 0.225$$

となった．結合演算のページオブジェクトに対するメソッド化により，ナイーブな実装と比較して約24倍の処理速度の向上が達成された．

4.5 関係代数演算の並列実行

関係代数演算内，演算間の並列実行の効果を調べるために，nCUBE2 を用いて選択演算内の並列実行および選択演算と射影演算のパイプライン並列実行の実験を行った．

演算内の並列実行を選択演算を例に，メソッドを呼び出しプロセスを複数の PE に割り当てる方針で実験してみたが，ほとんど効果はなかった．

次に，選択演算と射影演算のパイプライン並列実行の実験を行った．実験に用いたデータベースのサイズは10000タップルであり，SQL命令は以下の通りである．

```
SELECT table.unique1
FROM   table
WHERE  table.unique1 >= 0
AND    table.unique1 <= 10000
```

この命令を用いて，図11の3方式について比較を行った．

(1) 選択演算と射影演算を1PE上で実行する．演算間の並列性は得られない．

(2) 選択演算と射影演算は異なる PE 上で実行する．

(3) 選択演算，射影演算中の各マージャを異なる PE 上で実行する．

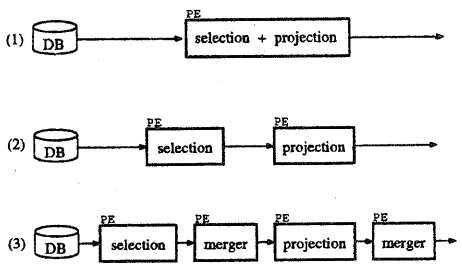

図 11: 演算間の並列実行

こららの結果はを表2に示す．なお，選択演算のみの実行時間は 7.04(s)，射影演算のみの実行時間は 19.30(s) であり，各演算に占めるディスクからのデータの読み出し時間は 4.40(s) である．

表 2: 演算のパイプライン実行 (単位: sec)

方式 (1)	方式 (2)	方式 (3)
22.36	19.49	18.56

方法 (2) の結果から，演算をパイプライン実行することにより並列性が得られているが，射影演算の実行時間により処理速度が抑えられてしまっている．

射影演算内のオーバヘッドの大きい処理を調べると，出力ページオブジェクトをまとめて出力ストリームにするマージャの処理であることが分かった．選択演算についても同様であったため，マージャの処理を異なる PE で実行する方式 (3) を試みた．この結果，方式 (3) では方式 (2) よりもさらに約 1(s) の時間短縮がなされた．

方式 (3) において，CPU モニタで実行の様子を観察したところ，各 PE がうまく並列動作しておらず，処理が逐次的になっている部分が多いことが分かった．これは，演算の出力ストリームがうまくパイプライン的にデータを転送していないことに原因があると考えられる．プロセスの優先度を変更してみたが，KLIC は優先度つきのプロセスを実行する場合，ソフトウェアオーバヘッドが大きくなるため，処理時間の短縮はできなかった．

KLIC のプロセスのスケジューリングは手続き駆動型であり，基本的にプログラムで記述されたプロセスの順序に従う．しかし，このスケジューリング方式では，データ要求に対するレスポンスは悪く，パイプライン実行に向かない．そこで，要求駆動型のスケジューリング方式による KLIC の実装方式が提案されている [10]．この要求駆動方式の KLIC を使用すれば，パイプライン実行がうまく動作すると考えられる．

5 おわりに

並列論理型言語 KLIC を用いてアクティブ関係データベース処理を行なう場合，ナイーブな実装では，I/O のオーバヘッド，GC のオーバヘッド，プロセススイッチのオーバヘッドが問題となる．これは，KLIC の扱うデータの粒度が小さく，全てのデータが GC の対象となるためである．

本論文では，KLIC のジェネリックオブジェク

トを用いて新しくページオブジェクトを導入し，これを用いてページ単位によるI/Oオーバヘッドの削減，GCのオーバヘッドの削減，ならびに関係代数演算のメソッド化によるリストを手繰るオーバヘッドとプロセススイッチのオーバヘッドを削減できることを述べた．

このページオブジェクトによりどれだけのオーバヘッドが削減できるかを，nCUBE2やWS上での実験を通して具体的に調べた．その結果，I/O性能で1/25~1/100，GCで1/400，関係代数演算で1/5~1/24にオーバヘッドを削減することが分かった．

以上の結果により，並列記述が容易な高級言語による低オーバヘッドの並列DBMSの可能性を示すことができた．

また，演算間の並列実行を試み，パイプライン並列性が得られることが確認できた．しかしながら，現在のKLICは手続き駆動型スケジューリングであるため，うまくパイプラインが動作していないことが分かった．

今後，要求駆動型のスケジューリング方式を採るKLICを用いて，演算間のパイプライン実行の効果を再検証する必要がある．また，関係代数演算のマージャのオーバヘッドが大きいため，出力ストリームの差分リストによる実装を行い，その効果を調べたい．さらに，データのパーティショニングを考慮にいれた演算の実装，たとえば，並列ハッシュジョインアルゴリズムを実装し，その効果を調べていきたい．

高級言語の並列化記述およびルール処理記述の容易さを生かしつつ，高速な処理を実現することが重要である．関係演算の要素となる基本モジュールの記述は，従来からのCのような言語で記述しても特に記述は困難でなく，性能を向上させることが可能である．基本モジュール間の並列動作やルール処理を，高級言語によって記述するアプローチが有効であろう．このようなアプローチはKLICに限らず，GCを伴うような並列言語にも適用できる．

謝辞

本研究の一部は，財団法人日本情報処理開発協会先端情報技術研究所および文部省科学研究費補助金重点領域研究「高度データベース」の助成により行なわれた．

参考文献

[1] J. Widom and S. Ceri: "Active Database Systems: Triggers and Rules For Advanced Database Processing", Morgan Kaufmann Publishers, INC., San Francisco, CA (1996).

[2] T. Fujise, T. Chikayama, K. Rokusawa and A. Nakase: "KLIC: A Portable Implementation of KL1", Proc. of FGCS '94, pp. 66–79 (1994).

[3] 久保, 麦, 宮崎, 横田："並列論理型言語を用いた並列アクティブデータベースシステムにおけるネステッドトランザクション管理", Proc. of DEWS '96 電子情報通信学会, pp. 67–72 (1996).

[4] 横田, 宮崎, 土屋："並列アクティブデータベースエンジンParadeの並列処理", 文部省科学研究費重点領域研究『高度データベース』松江ワークショップ講演論文集, 第2巻, pp. 426–433 (1996).

[5] 宮崎, 横田："並列アクティブデータベースParadeのルール処理機構", Proc. of DEWS '97 電子情報通信学会 (1997).

[6] 関田："Inside KLIC version 1.0", http: //www.icot.or.jp /AITEC /COLUMN /KLIC /inside / index.html (1998).

[7] 宮崎, 横田："並列論理型言語KLICによる並列データベースのI/O処理の高速化", 第55回全国大会 講演論文集 (分冊 3) 情報処理学会, pp. 461–462 (1997).

[8] J. Gray and A. Reuter: "Transaction Processing: Concepts and Techniques", Morgan Kaufmann Publishers, INC., San Francisco, CA (1993).

[9] 金政, 宮崎, 横田："並列データベースシステムにおける更新を考慮したディレクトリ構成", 信学技報, **DE97-77 (AI97-44)**, pp. 63–68 (1997).

[10] 近山, 宇佐："要求駆動型スケジューリングによるKL1実装方式の研究", http: //www.icot.or.jp /AITEC /FGCS /funding / itaku-H9-index-J.html (1998).