データベース・システム 60－4
（１９８７．７．２０）

マルチポートページメモリを用いた知識ベースマシンの並列制御方式と処理性能

物井 秀俊†、酒井 浩‡、伊藤 英則†
†(財)新世代コンピュータ技術開発機構
‡東芝総合研究所

大量の知識を格納管理する専用マシンとして、知識ベースマシン(KBM)を検討している。このKBMでは、並列実行可能な複数の検索専用プロセッサ(単一化エンジン)と知識を格納する二次記憶装置をマルチポートページメモリ(MPPM)で結合し、並列処理による高速な知識検索の実現を目指している。

KBMは再帰定義を許す関係型知識ベース(RKB)を検索の対象とする。RKBに対するキュエリ処理では、再帰定義を処理するため、RKB内の項関係に検索演算を繰り返し施すことが要求される。繰り返しを伴うキュエリ処理を並列処理により高速化するには、繰り返しの中で、各検索演算の実行結果を見ながら、次の検索演算の分割と並列実行のスケジューリングを、動的に制御する必要がある。このため、キュエリが決まると検索手順が固定的に決定するデータベース検索処理に対し、RKBのキュエリ処理では異なった観点からの効率化に関する考察が要求される。本稿では、上記のキュエリ処理を、提案しているKBMアーキテクチャ上で効率良く処理するための制御方式について考察する。特に、検索を並列実行させるため、演算を単純に分割して分配するとストリーム処理を行うUEへのデータ入力量が増加し、並列処理による処理効率を低下させることを示す。さらに、この分割損を低減させる制御方式について述べる。

Parallel Control Technique and Performance of an MPPM Knowledge Base Machine

Hidetoshi MONOI†, Hiroshi SAKAI‡, Hidenori ITOH†
†ICOT Research Center
4-28, Mita 1-Chome, Minatoku, Tokyo 108
‡Toshiba R & D Center
Toshibacho, Komukai, Saiwaiku, Kawasaki 210

We investigate a knowledge base machine (KBM) providing efficient management of large, shared knowledge base. This KBM is constructed from the multiport page-memory (MPPM) and unification engines (UEs), and executes each retrieval operation in parallel using multiple UEs for fast retrieve.

The KBM retrieves the relational knowledge base including recursive definitions, which cause repetition of the retrieval-by-unification (RBU) operations on term relations in the query process. For the parallel execution, the multiprocessor nested-loop algorithm is employed. Each RBU operation between two relations is divided into fine grain operations by partitioning input relations and the UEs are scheduled for concurrent execution.

The cost of query processing becomes large when the number of UEs grows so long as the multiprocessor nested-loop algorithm is employed. This paper proposes a new control strategy which exhibits stable performance when the number of UEs varies.

1.はじめに

高速処理を目指す人工知能向きマシンでは、処理対象となる知識を一次記憶上に展開する事を前提とした超並列マシンの研究が盛んである。このようなマシンでは、複雑な構造を持つ知識を効率良く処理するため、一次記憶を高機能化する傾向にある[1-3]。しかし、処理の対象となる知識が大量となった場合、全ての知識を高機能の一次記憶に格納することは、量的に不可能でありコスト面からも得策とは言い難い。従来のデータベースシステムのように、安価で大容量の二次記憶に知識を格納し、要求のある時だけ必要な知識を検索加工し、一次記憶に高速にロードするメカニズムが必要となる。

第五世代コンピュータプロジェクトでは、これらの要求を充たすため、大量の知識を管理格納する専用マシンとして知識ベースマシン(KBM : Knowledge base machine)を研究開発中である。第五世代コンピュータプロジェクトの前期では、KBMの第一ステップとして大量のデータを格納し管理するという立場から、関係データベースマシンDelta[4]を開発した。中期では、このDeltaで得られたノウハウを継承し、より高機能の検索処理が可能なKBMの開発を行っている。

二次記憶への知識の格納と検索処理の高速化を考えたとき、解決すべき課題として、以下のものが挙げられる。まず、知識の表現に関しては、ブロックアクセスという特性を積極的に生かすため、知識を集合のままで処理できる格納モデルの実現。また、KBMの構成としては、(1)二次記憶装置と検索プロセッサ間のデータ転送上の隘路を解消する結合方式の実現、さらに(2)大量の知識を効率よく処理する検索専用装置の実現である。

以上の課題に対し、我々は格納モデルとして関係型知識ベース(RKB : Relational knowledge base)を提案している。RKBでは、知識を項で表現し、項の集合を項関係(Term relation)とよぶ関係の形で格納する。また、項関係に対しては、関係代数を単一化操作(unification)により拡張した単一化検索演算(RBU演算 : Retrieval by unification operation)を用いる。RBU演算としては、単一化結合(unification join)や単一化制約(unification restriction)演算等がある[5]。さらに、KBMのアーキテクチャとして、マルチポートページメモリ(MPPM:Multiport page memory)[6]と検索専用プロセッサとしての単一化エンジン(UE:Unification engine)を用いた構成方式[5]、そしてストリーム処理を行うUEのハードウェアアルゴリズムとその実現方式を提案している[7]。

RKBでは項を直接格納するため、Prologで言うルールとファクトを同じ形式で格納することができる。このため、データベースのように格納したい実体(entity)全てを外延的(extensional)に定義するだけでなく、ルールによって再帰定義を含む内包的(intensional)な定義が可能となる。

RKBに対するキュエリ処理では、実体に対する柔軟な定義が可能であるため、検索処理自体に内包的な定義から全ての解を展開する機能が求められる。特に、再帰的な定義からの解の導出は、RKB内の項集合に繰返しRBU演算を施すことが必要となる。この繰返しを伴うキュエリ処理を、複数UEを用いて並列処理するには、繰り返しの中で各RBU演算の実行結果を見ながら、次のRBU演算の分割と並列実行のスケジュールを行うという、動的な制御が必要となる。データベースのキュエリ処理では、各キュエリに対し検索手順が固定的に決定されるため、並列処理に対して種々の効率化が考えられている[18][19]。しかし、RKBに対するキュエリ処理のように動的な制御を必要とする場合は、効率化についてあまり検討されていない。

本稿では、上記のようなRKBのキュエリ処理に対して、提案しているKBMアーキテクチャ上で効率良く実行するための制御方式と、シミュレーションによる評価結果について述べる。まず、演算を単純に分割して並列度を増すと、UEへのデータ入力量が増加して、並列処理による処理効率を低下させることについて述べ。さらに、この分割損を低減させる制御方式について検討する。以下では、2章で制御の対象となるKBMのアーキテクチャと主要な構成要素について述べる。次いで3章では、並列実行のためのRBU演算の分割方式と制御方式について述べる。4章では、3章で述べた制御方式に対するシミュレーション結果について述べ、最後に5章でこのシミュレーション結果から得られた問題点と今後の課題について述べる。

2.知識ベースマシンアーキテクチャ

2.1 全体構成

KBMの全体構成を図2-1に示す。制御プロセッサ(CP : Control processor),マルチポートページメモリ,単一化エンジンおよび二次記憶装置(DKS : Disk system)を、主な構成要素とする。

関係型知識ベースは、DKSに格納される。UEは、項関係に対してRBU演算を実行する、専用プロセッサである。図2-1に示すように、UEとDKSはMPPMを介して結合する。DKS中の項関係は、まずMPPMにステージングする。UEは、このMPPM上にステージングした項集合を直接入力して、検索処理を実行する。CPは、KBM全体の制御とKBM/ユーザ間のインタフェース処理を司る。

この構成の目的は、(1)MPPMをDKSに対するキャッシュメモリとして用い、かつDKSとUEを複数のポートで結合してデータ転送上の隘路を解消す

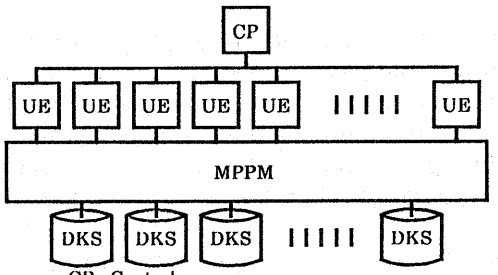

図2-1 Knowledge base machine configuration

ること、(2)MPPMを複数のUEから同時アクセス可能な共有メモリとして使用し、複数UEを並列実行させる場合のデータ転送隘路を解消することにある。

2.2 マルチポートページメモリ

本稿で述べるKBMに期待する処理は、データベースマシンと同様に大量のデータを対象とする処理の単純な組み合わせである。このため、アーキテクチャとしては、検索プロセッサの高速化のみならず、二次記憶装置からディスクキャッシュおよびディスクキャッシュから検索プロセッサへの高速なデータ転送路の確保が要求される。特に、本稿で述べるKBMでは、個々の検索演算の入力を分割して複数のUEにより並列実行することを考えており、ディスクキャッシュには、複数UEから同時アクセス可能な、共有メモリとしての機能が求められる。

以上の観点から、ディスクキャッシュメモリと複数UEとの結合方式として、共有バス(common bus),クロスバ(crossbar switch)及びMPPMの3方式を取り上げ、転送路の競合に着目して処理能力の比較を行った。ここで、各方式におけるバスの転送能力としては、共有バス方式では50MB/sec, 100MB/sec, 150MB/secを、クロスバ方式とMPPM方式については5MB/sec, 7MB/sec, 10MB/secを仮定した。ここで、クロスバ方式では、各UEからのディスクキャッシュ上の同一メモリモジュールへのアクセス競合が問題となる。そこで、この確率を0.1と仮定した。シミュレーションによる結果を図2-2に示す。

MPPMは、各プロセッサに対して常に独立した転送路が与えられるため、データ転送上の競合が無く、プロセッサ台数に比例して処理能力が向上する。これに対して、共有バス方式では、全てのプロセッサに対するデータ転送が一つのバスを通して行われる。このため、かなり転送能力の高いバスを仮定しても、バス上の競合により処理能力が頭打ちとなる。また、クロスバ方式では、同一メモリモジュールへのアクセスがあると複数の転送路を確保することができない。このため、共有バス方式ほ

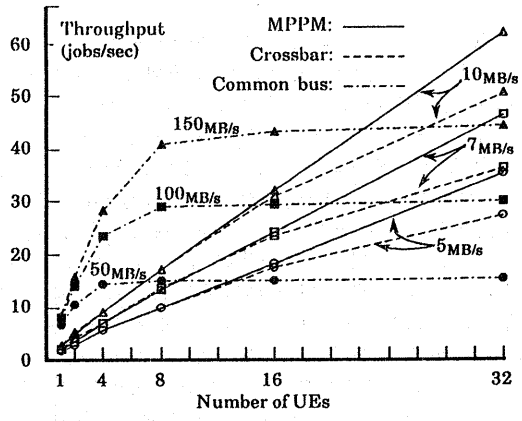

図2-2 Comparison of interconnection structures

どではないが、UE台数が増加して同じメモリモジュールへのアクセスが多くなると転送路上の競合が発生し、処理能力が頭打ちとなる。

このほか、結合方式ではハードウェア量の問題も考える必要があるが、比較的単純なネットワーク構成でMPPMを実現できる[6]。

2.3 単一化エンジン

UEは、項集合を入力してRBU演算を高速に実行する専用プロセッサである。二次記憶上に格納された項集合を効率良く処理するため、項集合を直接入力するストリーム処理を実現している。

UEの構成を図2-3に示す。2つの入力ポートと1つの出力ポートを持ち、2ウェイマージソートアルゴリズムにより項をソートするソータ部(sort unit),単一化の可能のあるタップルの組を作るペア生成部(pair generation unit),そしてペア生成部が出力した項の組に対して単一化を実施する単一化処理部(unification unit)から構成される[7]。

ソータを使用しているため、UEがストリームとして一度に処理できる項集合の最大量は、このソータの容量になる[8]。この容量は、UEで実行できる問題の上限値を規定するものであり、RBU演算を分割する場合に、重要なパラメタとなる。以下では、この最大量をUEのバッファサイズとよぶ。

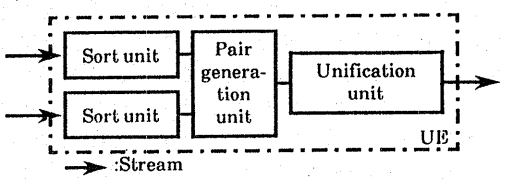

図2-3 Configuration of UE

3.知識ベースマシンにおける検索処理

3.1 関係型知識ベースと検索処理

RKBに対するキュエリ処理は、1章で述べたように、RKB内の項関係に対しRBU演算を繰返し施すことにより実現される。キュエリ処理の例として、

Prologのプログラムをリスト構造で表現し、headとbodyという2属性の項関係(*Pr*)に格納した場合の検索手順を、図3.1に示す[5]。図3.1では、Prologプログラムに与えられるゴールがキュエリとなる。

処理は以下のようになる。まず、知識ベース内の項関係*Pr*から、body属性がゴール(*goal*)と単一化可能なタップルを抜き出し、項関係Tr_0を作る。このTr_0と*Pr*との間で単一化結合を実行して項関係Tr_1を作る。項関係Tr_iと項関係*Pr*との間の単一化結合により新しい項関係Tr_{i+1}を作る操作を、新たな項関係ができなくなる($Tr_i=\phi$)まで繰り返す。また、解となる項集合*R*は、各Tr_iからbody属性が空リスト(*body*=[])となったタップルを集めて求める。

以下でキュエリ処理という場合は、図3-1に示す単一化結合の繰り返しによる検索手順を指すことにする。また、項関係*Pr*は、このキュエリ処理の間変更されることがないため永久項関係とよび、単一化結合の出力である各Tr_iを一時的項関係とよぶ。さらに、1回の単一化結合処理を世代と呼び、*Pr*とTr_iからTr_{i+1}を作る単一化結合処理を第i世代とよぶ。

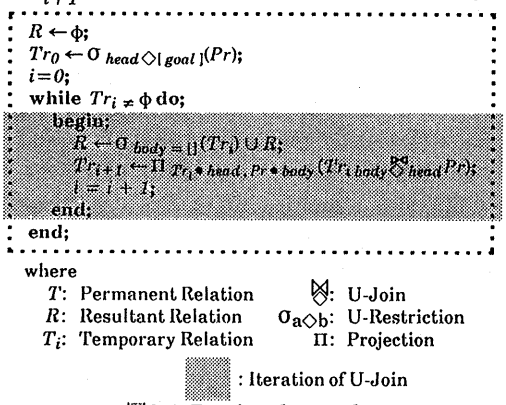

図3-1 Retrieval procedure in the relational knowledge base

3.2 キュエリ処理とUEの並列実行

単一化結合や単一化制約などのRBU演算は、粒度の大きな演算であり、並列処理による効果を上げるには、入力データを分割して複数のプロセッサに割り付ける問題分割が必要となる。ここで、RBU演算は単一化を条件とする関係演算であり、実行時間が入力データの内容に大きく依存する[16]。このため、入力データを量的に均等に分割し複数UE並列実行してもUE毎に処理時間が異なり、演算を分散させたUE間で同期を取ると、UEの待ち時間が多くなる場合がある。以下で述べるUEの並列実行方式では、項集合を分割して分割単位毎にRBU演算を生成する。そして、分割したRBU演算毎にUEに割り当て、UE間の同期を不要にする。

本稿のKBMでのキュエリ処理は、図3-2に示すようになる。先ず、KBMに到着したキュエリは、RKB内の項関係とRBU演算からなるコマンド列にコンパイルされる。コマンド列内の各RBU演算は、MPPMのページを単位とする処理に分割され、UEに割り当てられる。各UEは、入力となるページおよび演算結果を出力するページが決定すると、入力ページと出力ページの間にストリームを形成する。そして、ストリームを流す間にRBU演算を実行する。UEが形成するストリームには、単一化結合演算等に対する2入力1出力のものと単一化制約演算等に対する1入力1出力という2種類がある。

RKBにおけるキュエリ処理で、単一化結合の繰返しの度に項関係をDKSからアクセスすると、DKSからのデータ転送が多くなる。アクセスする項関係はキュエリ処理が完了するまでMPPM上に格納し、DKSに対する頻繁なアクセスを回避する。また、個々の演算の入力を分割して並列実行する場合、同じ分割単位に対するアクセスが多くなる。しかし、2.2節で述べたように、MPPMを用いた場合アクセス競合は問題とならない。

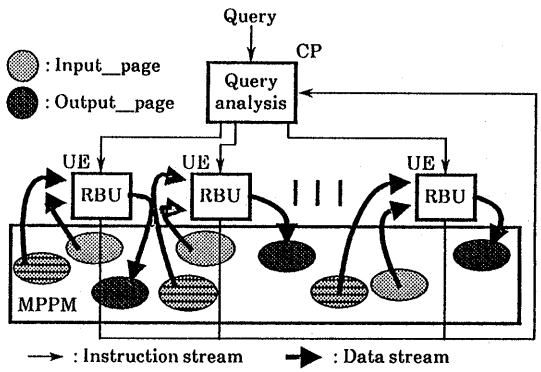

図3-2 Query processing in the KBM

3.3単一化検索演算の並列処理方式

3.3.1 単一化検索演算の分割方式

検索演算を並列実行させるときの、演算の分割方式について考える。以下では、検索演算の分割処理を問題分割(division process)と呼び、分割された個々の演算を部分問題(split operation)と呼ぶ。

関係演算の並列実行方式、特に関係同士の結合演算の並列実行に関する研究は、マルチプロセッサデータベースマシンにおいて盛んに行われている[9][10]。本稿では、図3-3に示すように、2つの項関係をタップル方向に分割して並列実行可能な部分問題を作成する、マルチプロセッサネステッドループアルゴリズム[11]について考える。項関係の分割はMPPMのページサイズを単位とし、分割した個々のタップル集合をセグメントとよぶ。

この分割を行った場合、部分問題は以下のように

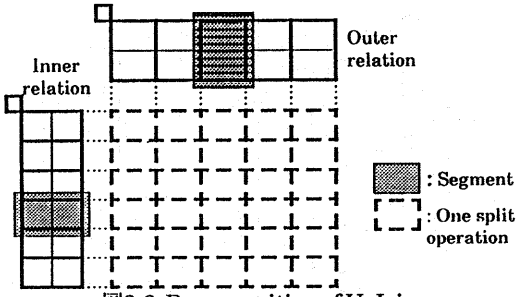

図3-3 Decomposition of U-Join

なる。今、項関係Tの大きさを$|T|$で表し、セグメントサイズをsとしてタップル方向に(一つのタップルを途中から分割しないように)分割した場合のセグメント数を$|T|//s$と書く。また、Tを分割してできるセグメントの集合を$\{T\}$と書く。項関係Q,R間の単一化結合で、セグメントサイズを各々s_1,s_2とすると、部分問題の入力となるセグメントの全組み合わせは、以下のようになる。

$$\{(q_i, r_j) \mid q_i \in \{Q\}, p_j \in \{R\}\}$$

ここで、$1 \leqq i \leqq |Q|//s_1, 1 \leqq j \leqq |R|//s_2$

次に、上記の部分問題をUEで実行した場合の、入力量について考える。入力量を考えるのは、ストリーム処理をUEで実行する場合、演算の処理量が入力量によって決まるためである。UEは、図2-3に示すように2つの入力ポートを持ち並列読み込みを行う。このため、$s_1 \geqq s_2$とすると、s_1がUEの実行時間として表に現れる入力量となる。単一化結合を分割して実行した場合、各部分問題の実行でセグメントの入力を行うため、総入力量I_Dは

$$\begin{aligned} I_D &= s_1 \times (|Q|//s_1) \times (|R|//s_2) \\ &= |Q| \times (|R|//s_2) \end{aligned}$$

$//$の定義より$s_1 \times |Q|//s_1 = |Q|$

となる。これに対し、QとRを分割せずに一つのUEで単一化結合を実行した場合の入力量Iは、$|Q| \geqq |R|$とすると、QとRを一度だけ入力すれば済むため$I = |Q|$となる。IとI_Dを比較すると、I_DはIの$|R|//s_2$倍となる。これは、入力量が分割数に比例して増加することを意味する。

以上のように、結合演算の入力となる関係を単純に分割して部分問題を生成すると、分割数に比例して入力量が増加する。入力量の増加は、ストリーム処理を行うUEを用いた場合、処理量の増加となる。この問題は、関係データベースの結合演算でも同様であり、ハッシングによってセグメントの組合せを削減する方式が提案されている[12][13]。

以下では、キュエリ処理を複数UEで並列実行した場合の効果を明確にするため、ハッシング等の方式について言及することは避ける。結合演算の繰り返し処理における制御方式について議論する。

ここで述べた分割方式を、図3-2に示すキュエリの実行方式に適用すると、図3-4に示す制御手順となる。先ず、キュエリに対して問題分割を行なって部分問題を生成し、各部分問題をUEに割り当てる。UEによる部分問題の実行が完了すると、そこで出力されたタップルの集合と永久項関係との間で新たな問題分割を行ない、生成した部分問題をUEに割り当てる。この問題分割と部分問題の実行を、UEからの新たな出力が無くなるまで繰り返す。

我々は、問題分割の方式に対して、セグメントの大きさを固定して常に一定の大きさの部分問題を生成するSP(single page at a time)方式と、並列実行するUEの台数を固定しUEから出力されたタップル集合の大きさに応じて部分問題の大きさを調整するMP(multiple page at a time)方式を考えた。次に、この2つの方式について述べる。

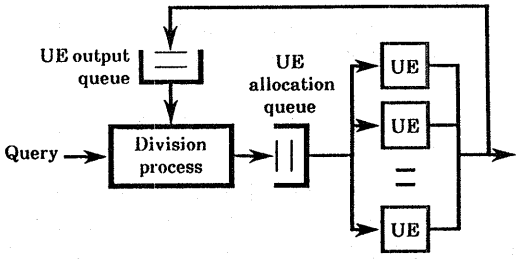

図3-4 Control for parallel execution of the query processing

3.3.2 SP方式

RKBのキュエリ処理で、単一化結合を実行する度に、単一化結合を分散させた全UEの同期を取ると、待ち時間が多くなりUEの稼働率を低下させる可能性がある。そこで、問題分割のセグメントサイズをMPPMのページサイズに固定し、各UEの単一化結合処理が終了した時点でUE毎に部分問題を生成する、データ駆動的な方式を考える。以下では、この問題分割方式をSP方式とよぶ。

SP方式によるキュエリ処理は、以下のようになる。永久項関係Prと一時的項関係$Tr_i(i \geqq 0)$をMPPMのページサイズpで分割すると、永久項関係Prと一時的項関係の初期値Tr_0は、各々

$$\{Pr\} = \{t_i \mid 1 \leqq i \leqq |Pr|//p\}$$
$$\{Tr_0\} = \{t_{0,i} \mid 1 \leqq i \leqq |Tr_0|//p\}$$

ここで、t_iと$t_{0,i}$は各々PrとTr_0のセグメント

というセグメントの集合に分割される。この分割によって、PrとTr_0間の単一化結合は、

$$t_i \overset{\bowtie}{\diamond} t_{0,j}$$

ここで、$1 \leqq i \leqq |Pr|//p$、$1 \leqq j \leqq |Tr_0|//p$

という部分問題の形で各UEに分配される。

次に、第k世代($k \geqq 1$)の単一化結合について考える。UE_1,---,UE_nというn台のUEを使うとして、第$k-1$世代の単一化結合に対するUE_mの出力をu^m_{k-1}と表す。第k世代の単一化結合は、第$k-1$世代の単一化

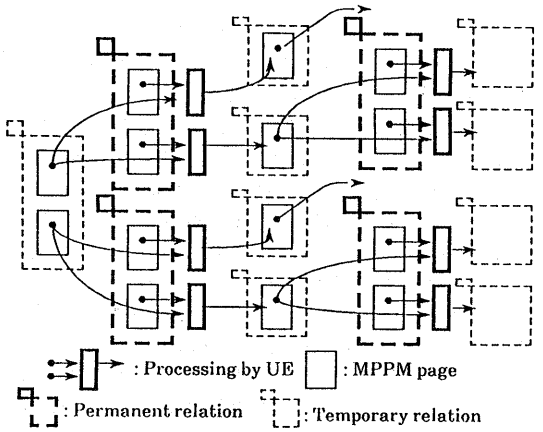

図3-5 Parallel execution in SP method

結合に対する各UEの出力を、独立にページサイズで分割したものである。第k世代の単一化結合に対して生成される部分問題の総数は、

$$\sum_{m=1}^{n}(|Pr|//p \times |u_{k-1}^{m}|//p))$$

となる。

SP方式では、図3-5に示すように各世代毎の部分問題の生成をデータ駆動的に行うため、オーバヘッドの少ない制御方式が実現できる。しかし、分割の対象となる項関係の大きさに対してMPPMのページサイズが小さすぎると、項関係の分割が細かくなり処理量を増加させる。一方、MPPMのページサイズが大きすぎると生成する部分問題の数が少なくなり、UE台数に対して充分な並列度が得られない。SP方式では、全ての問題に適合するMPPMのページサイズを求める事が非常に難しくなる。

3.3.3 MP方式

MP方式では、セグメントの大きさを、並列実行するUE台数により調整する。但し、MPPMはページ単位のアクセスしかできないため、セグメントの大きさは、MPPMのページサイズで量子化する。この方式では、並列実行させるUEの台数(以下でこれを並列度と呼ぶ)を予め決めておき、問題分割に際しては常に並列度分の部分問題を生成する。

並列度を固定して問題分割を行う場合、SP方式のようにUE毎に問題分割を行うと、並列度によっては非常に細かい分割となる。そこで、セグメントの大きさを調整できるように、各UEで出力する部分問題の実行結果(タップル集合)を一か所に集め、このタップル集合(これが、キュエリ処理の一時的項関係となる。但し、格納するUEの出力は、異なる世代のものを同居させる。)と永久項関係の間で問題分割を実施する。問題分割を開始する契機は、一時的項関係の大きさがUEのバッファサイズ以上になるか空きUEができた時点とする。空きUEの検出を問

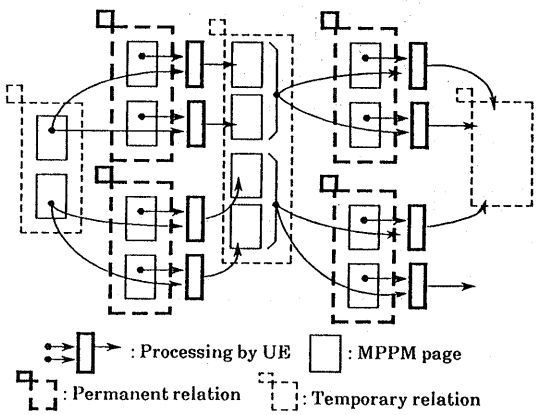

図3-6 Parallel execution in MP method

題分割の契機とするのは、UEの稼働率の向上が目的である。MP方式による実行手順を、図3-6に示す。

MP方式では、以下のような手順により問題分割のセグメントサイズを決定する。

いま、分割を行う時点での一時的項関係をTrとし永久項関係をPrとする。また、PrとTrを各々l,m個のセグメントに分割するとし、この時のセグメントの大きさを各々s_p,s_tとする。MP方式では、並列度分の部分問題を生成するため、並列度をnとすると$l \times m = n$である。問題分割で処理量を出来るだけ増加させないため、部分問題を実行した場合の総入力量を少なくするようにs_pとs_tを決める。総入力量I_Dは、並列度nに対し以下のように計算できる。

$$\begin{aligned} I_D &= n \times \max(|Pr|//l\ ,|Tr|//m) \\ &\geqq n \times \{(|Pr|//l + |Tr|//m)/2\} \\ &\qquad \text{ここで、等号は} s_p = s_t \text{のとき} \\ &= (m|Pr| + l|Tr|)/2 \\ &\geqq (m|Pr| \times l|Tr|)^{1/2} \\ &\qquad \text{ここで、等号は} s_p = s_t \text{のとき} \\ &\geqq \{n(|Pr| \times |Tr|)\}^{1/2} \end{aligned}$$

計算過程から、I_Dが最小になるのは$s_p = s_t$の場合である。MP方式では、$|Pr| \times |Tr|$をn個に等分割しその平方根をs_p,s_t(即ち、$s_p = s_t = \{(|Pr| \times |Tr|)/n\}^{1/2}$)として、セグメントの大きさを求める。

しかし、UEのバッファサイズをbとすると、$s_p \leqq b$かつ$s_t \leqq b$でなければならない。さらに、問題分割を単純にするため、各問題分割でPrを使い切る必要がある。$|Pr| \times |Tr| \geqq (n \times b^2)$の場合(特に、$|Pr| \geqq n \times b$の場合)は、$s_p = b$として$n$個以上の部分問題を生成する。この時、$s_t$については、$|Tr| \geqq b$であれば$s_t = b$として$Tr$の残りは次の問題分割にまわす、また$|Tr| < b$であれば$s_p = |Tr|$とする。

以上の分割処理を、各UEからの新たな出力が無くなるまで繰り返す。

MP方式では、セグメントサイズの調整を問題分

割の度に行う必要があり、SP方式に比べて問題分割に時間かかるようになる。

4.処理性能

以下では、SP方式とMP方式を図2-1に示すKBMで実行した場合の処理性能について述べる。

結果は、全てシミュレーション[15]によるものである。シミュレーションでは、図3-4に示す制御手順をモデルとした。また、UEについては、[7]で提案したUEをクロック単位でシミュレートして[14][16]、実行時間に関しては、実機を作った場合と正確に一致する値を出している。MPPMの容量については制限をせず、二次記憶との入出力を考えない状態でキュエリ処理を実行している。さらに、分割方式の影響を明確にするため、キュエリのコンパイル,問題分割,部分問題の管理等の制御上のオーバヘッドは処理時間には加えていない。

4.1 SP方式の性能特性

SP方式による処理時間とUEの稼働率を、ページサイズ毎に求めた結果を図4-1に示す。

SP方式で生成される部分問題の数はページサイズに従う。このため、UE台数が同じ場合は、ページサイズが大きいほど分割損による入力量の増加が少なくなり、処理時間が短くなる。しかし、ページサイズを大きくし過ぎるとUE台数分の部分問題が生成されなくなり、UEの稼働率が低下して処理時間を悪くする。また、UE台数が多い場合は、問題の分割による入力量の増加に対して並列処理の効果が勝り、処理時間の変化は少なくなる。

上記の処理時間の変化は、UE稼働率で裏付けることができる。即ち、ページサイズが小さくてUE台数が少ない場合は、UE台数に対して充分な部分問題があるため、高い稼働率となっている。しかし、ページサイズが大きい場合は、問題の分割が粗くUE台数に対して充分な部分問題を生成できないため、UE台数が多くなると稼働率が低下する。

以上のように、UE台数や項関係の大きさといった実行環境に対応して、処理時間を最小にするセグメントサイズの最適値が存在する。SP方式のようにセグメントサイズを固定する方式では、全ての動作環境に適合するセグメントサイズを選定することが難しくなる。

4.2 MP方式の性能特性

MP方式では、問題分割の並列度をUE台数と等しくしてシミュレーションを行った。MP方式による処理時間と稼働率のグラフを図4-2に示す。

SP方式に対して、ページサイズが小さいときの処理時間が、非常に安定し且つ短くなっている。これは、MP方式では常にUE台数分の部分問題を生成するため、SP方式のように無駄な問題分割をすることが無くなったためである。

稼働率については、UEの出力を溜めてから問題分割を行うため、UEの待ち時間が増加しSP方式に比べて稼働率が全体的に低くなっている。しかし、この稼働率の低下は、処理時間を極端に悪くするほどにはなっていない。また、ページサイズが大きくなると稼働率が低下するが、これはSP方式の場合と同じ理由による。

5.議論

以上、RKBに対するキュエリ処理を効率的に実行するための制御方式について述べた。検索演算を並列実行するときの問題分割方式として、SP方式とMP方式という2つの方式を考え、シミュレーショ

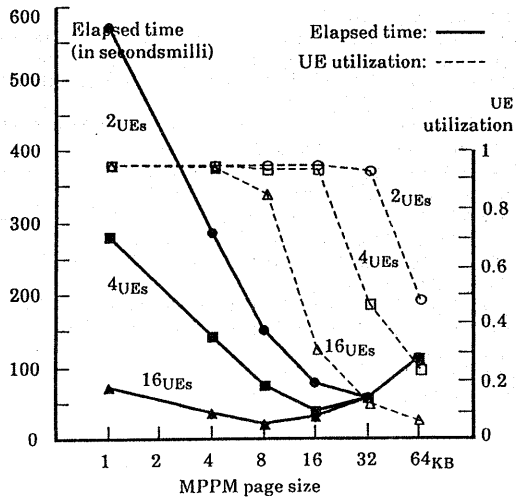

図4-1 Elapsed time and UE utilization in SP method

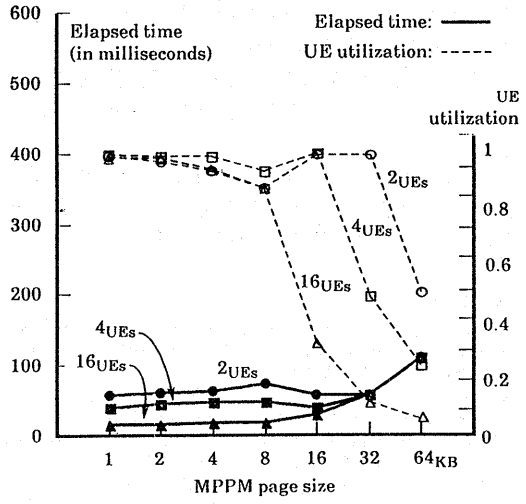

図4-2 Elapsed time and UE utilization in MP method

ンによる評価を行った。

本稿で述べたような単純な分割方式では、問題分割を細かくするほどUEへの入力量が増加し、処理効率を悪くする。特に、並列度が小さい場合、SP方式のように並列度と無関係に部分問題を生成すると、並列処理の効果が殆ど無くなってしまう。これに対して、並列度に応じて問題分割を行うMP方式では、並列度が小さい場合の無駄な問題分割が無くなり、SP方式に比べて処理時間が格段に向上する。しかし、MP方式でも、問題分割による入力量の増加を完全に除くまでは至っていない。例えば、図5-1に示すMP方式の台数効果(ここでSpeed up=[並列度nの処理時間]/[並列度1の処理時間])を見ると、性能向上率は並列度に比例していない。これは、MP方式では並列度が増加すると部分問題の数が増加し、並列度の増加が入力量の増加となって、キュエリ処理の処理効率を低下させているためである。

MP方式で、並列度が大きい場合の処理効率を上げるには、問題分割による分割損を減少させることが必要になる。本稿で述べた問題分割方式では、常に分割したセグメントの全組み合わせを部分問題として生成している。ここで生成した部分問題の中には、単一化で結合できるタップルの組み合わせがないものもある。問題分割に際して、このような無駄な部分問題の生成を削減できれば、かなりの入力量の削減が可能となる。関係データベースの結合演算では、ハッシュ値によってセグメントの分割を行い、同じハッシュ値を持つセグメント(クラスタ)どうしのみを組み合わせて部分問題とする問題分割方式により、無駄な部分問題の生成を防いでいる[12][13]。RKBにおけるキュエリ処理でも、[17][18]で提案している項に対するインデックス手法を用いることにより、項集合の分割とセグメントの組合せの省略が可能と思われる。

本稿で述べたキュエリ処理の並列制御方式への、インデックス手法の導入は、今後の課題である。

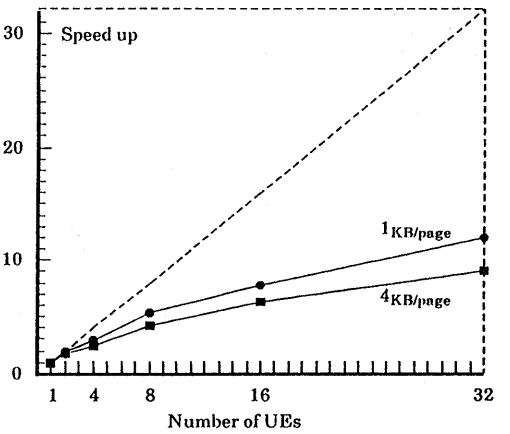

図5-1 Effect of parallel processing.

[謝辞]

本検討を進めるに当たり、有益な御示唆を頂いたICOT第三研究室の方々に感謝致します。また、熱心な討論を頂いた東芝のVLKB会議メンバの方々に感謝いたします。

[参考文献]

[1] Onai, R., et.al. "Architecture of a Reduction-based Parallel Inference Machine : PIM-R", *New Generation Computing*, OHMSHA, 3(2), June 1985.

[2] Ito, N., et.al., "The Dataflow-based Parallel Inference Machine to Support Two Basic languages in KL1", In *Proc. IFIP TC-10 Working Conference of Fifth Generation Computer Architecture*, UMIST (Manchester), July 1985.

[3] Moto-oka, T., et.al., "The Architecture of a Parallel Inference Engine-PIE-", In *FGCS '84*, ICOT, November 1984.

[4] Kakuta, T., et.al., "The Design and Implementation of Relational Database Machine Delta", In *Proc. Int. Workshop on Database Machine '85*, March 1985.

[5] Yokota, H., et. al., "A Model and an Architecture for a Relational Knowledge Base", In *Proc. 13th Ann. Int. Symp. Computer Architecture*, June 1986, pp.2-9.

[6] Tanaka, Y., "A Multiport Page-Memory Architecture and A Multiport Disk-Cache System, *New Generation Computing*, OHMSHA, 2, February 1984, pp.241-260.

[7] Morita, Y.,et.al., "Retrieval-By-Unification Operation on a Relational Knowledge Base", In *Proc. 12th Int. Conf. VLDB*, August 1986, pp.52-59,.

[8] Itoh, H., et.al., "Parallel Control Technique for Dedicated Relational Database Engines", In *Proc. 3rd Int. Confe. on Data Engineering*, February 1987, pp.208-215.

[9] Valduriez, P., "Semi-Join Algorithms for Multiprocessor Systems",*ACM TODS*, Vol.9, No.1, 1984, pp.133-161.

[10] Boral, H., et.al., "Processor Allocation Strategies for Multiprocessor Database Machines", *ACM TODS*, Vol.6, No.2, June 1981, pp.227-254.

[11] Boral, H., et.al., "Design Consideration for Data-flow Database Machines", In *Proc. ACM-SIGMOD 1980 int. Conf. Management of Data*, May 1980, pp.95-104.

[12] Kitsuregawa, M., "Architecture and Performance of Relational Algebra Machine GRACE', *University of Tokyo, Technical Report*,1983.

[13]DeWitt, D., "Multiprocessor Hash-Based Join Algorithms", In *Proc. Int. Conf. VLDB*, 1985, pp.151-164.

[14]伊藤,他,「大規模知識ベースマシンの開発(1)-(4)」,33会情処全国大会予稿集,1986.

[15]柴山,他,「大規模知識ベースマシン実験機の開発(2)~(3)」,34会情処全国大会予稿集,1987.

[16]森田,他,「知識ベースマシンにおける単一化専用装置の処理方式とその評価」,情処研究会 DBS57-4, 1987.,

[17]森田,他,「スーパーインポーズドコードを用いた構造体の検索方式」,33会情処全国大会予稿集,1986.

[18]大森,他,「推論機能と関係データベースの融合方式」,信学技報 AI86-21, 1986.

[19] Kiyoki, Y., "A Relational Database Machine Based on Functional Programming Concepts", In *Proc. FJCC* , 1986, pp.969-978.

[20]喜連川,他,「データベースマシン」,情報処理, Vol.28, No.1, pp.56-67, 1987.