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†† 森田幸伯†† 伊藤英則††
岩田和秀†† 酒井浩††† 柴山茂樹†††

項で表現された大量の知識を格納管理する知識ベースマシン (KBM) の，並列制御方式について考察する．KBM 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，並列処理により知識ベースの検索処理を高速に実行する．KBM は再帰的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，再帰的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，処理効率を低下させることを示す．さらに，問題分割時の入力量の増加を低減させる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．

1. まえがき

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こと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，すべ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，検索要求のある時だけ必要な知識を加工し一次記憶に高速にロードするメカニズムとして，知識ベースマシン (KBM : Knowledge base machine) が必要となる[4)]．

二次記憶への知識の格納と二次記憶からの知識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，①構造を持つ知識表現データを平坦な記憶構造しか持たない二次記憶装置へ記憶する格納形式の実現，②知識を集合のままで処理し，ブロック単位のアクセス特性を積極的に生かす格納モデルの実現．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あい路 (bottleneck) を解消する結合方式の実現，さらに②大量の知識を効率良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，格納モデルとして関係型知識ベース (RKB : Relational knowledge base) を提案した．RKB では，項で表現される知識を対象とし，項の集合を関係の形で格納する．この関係を項関係 (term relation) と呼ぶ．項関係に対しては，関係代数の等号条件を単一化操作 (unification) に拡張した，単一化検索演算 (RBU 演算 : Retrieval by unification operation) を用いる．RBU 演算には，単一化結合 (unification join) や単一化制約 (unification restriction) 演算等がある[5)]．また，KBM のアーキテクチャとしては，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 は項を関係の属性値とするため，外延的 (extensional) な定義 (Prolog プログラムのファクト) だけでなく，再帰的定義を含む内包的 (intensional) な定義（ルール）も，一つの項関係で格納することがで

† Parallel Control Technique and Performance of a Knowledge Base Machine Using Multiport Page-memory by HIDETOSHI MONOI, YUKIHIRO MORITA, HIDENORI ITOH, KAZUHIDE IWATA (Institute for New Generation Computer Technology), HIROSHI SAKAI and SHIGEKI SHIBAYAMA (Toshiba Research and Development Center).

†† (財)新世代コンピュータ技術開発機構
††† (株)東芝総合研究所

きる．内包的な定義を項関係の中に含むため，RKBのキュエリ処理では，内包的な定義からすべての解を探索する処理が求められる．特に，再帰的な定義からの解の導出には，項集合にRBU演算を繰り返し施す処理が必要となる．

単一化を伴う演算は，処理時間や結果のデータ量等の予測が一般には難しい．このため，繰り返しを伴うキュエリ処理を複数UEを用いて効率良く並列処理するには，繰り返しの過程で，各段階の演算の実行結果を見て次の演算の分割と並列実行のスケジュールを行う，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できるため，並列処理に関して種々の効率化が考えられている[19),20)]．しかし，検索演算の繰り返しの中で動的な制御を必要とする場合，並列処理の効率化は検討されていない．

本稿では，上記のようなRKBのキュエリ処理を，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，そのシミュレーションによる評価結果について考察する．以下では，2章で本稿で対象とするKBMのアーキテクチャとその主要な構成要素について述べる．3章では，まず演算を分割して並列実行する場合の問題点について考察し，さらに並列実行制御方式について検討する．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考察し，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 全体構成とキュエリの並列実行方式

KBMの全体構成を図1に示す．KBM全体の制御とKBMとユーザ間のインタフェース処理を司る制御プロセッサ (CP: Control processor)，ページをアクセス単位として複数ポートからの同時アクセスを許すMPPM，項の集合に対してストリーム処理を実行するUE，およびRKBを格納する二次記憶装置 (DKS: Disk system) を，主な構成要素とする．

このKBMでのキュエリ処理を，図2に示す．まず，KBMに到着したキュエリを，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する．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，各UEに割り当てられる．UEは，入力ページおよび演算結果の出力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して，割り当てられたRBU演算を実行する．

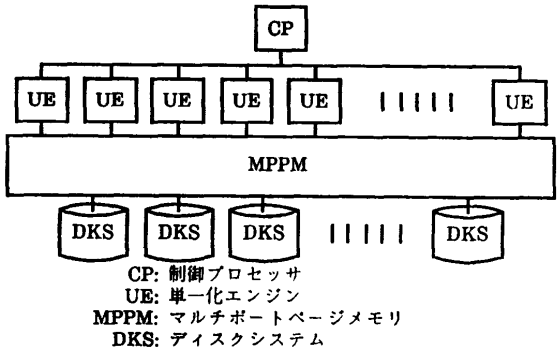


図1 知識ベースマシンの構成
Fig. 1 Knowledge base machine configuration.

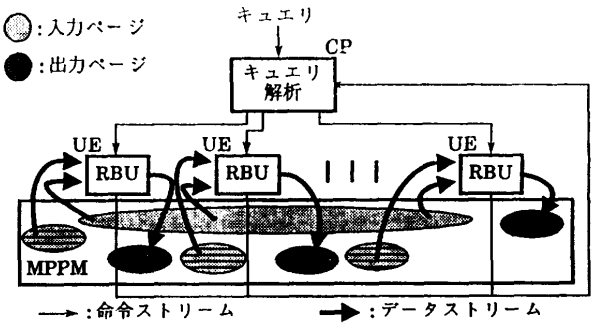


図2 KBMにおけるキュエリ処理方式
Fig. 2 Query processing in a KBM.

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かつDKSとUEを複数のポートで結合して，DKSとUE間のデータ転送上のあい路を解消すること，(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，複数UEを並列実行させる場合のMPPMとUE間のデータ転送あい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に要求される処理は，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演算の単純な組み合わせである．このため，アーキテクチャでは，検索プロセッサであるUEの高速化のみならず，DKSとディスクキャッシュおよびディスクキャッシュとUE間の高速なデータ転送路の確保が要求される．また，個々の演算の入力を分割して並列実行するため，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も求められる．

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) および MPPM の3方式を取り上げ，転送路の競合とメモリ上のアクセス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は，共有バス方式では 50 MB/sec, 100 MB/sec, 150 MB/sec を，クロスバ方式と MPPM 方式では 5 MB/sec, 7 MB/sec, 10 MB/sec を仮定した．さらに，クロスバ方式については，各UEからディスクキャッシュをアクセスしたとき，他のUEとの間でディスクキャッシュ上のアクセス競合が起こる確率を 0.1 と仮定した．結合するUEの数を変化させて，以下のような3種類のジョブをランダムに投入したときのスループットを，シミュレーションにより求めた．結果を**図3**に示す．

	入力データ量	UE 処理時間	出力データ量
A	1 MB	1 ms	1 MB
B	2 MB	2 ms	1 MB
C	2 MB	3 ms	2 MB

MPPM は，各 UE 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，UE 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UEに対するデータ転送が一つのバスを通して行われるため，バスの競合により処理能力が頭打ちとなる．また，クロスバ方式でも，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ず，UE 台数が増加するとアクセス競合により処理能力が頭打ちとなる．

このほか，MPPM はアクセスをページ単位としていることにより，比較的単純なネットワーク構成で実現できる[6]．結合方式を考える場合，ハードウェア量の問題も考える必要があるが，本稿では考察しない．

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4**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，単一化の可能性のあるタップルの組を作るペア生成部，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 が演算を実行する際に一度に入力できる量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，この量をUEのバッファサイズと呼び C_{bf} と書く．

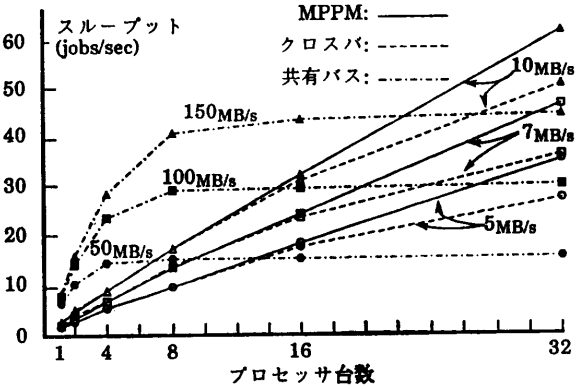


図3 結合方式による性能比較
Fig. 3 Comparison of interconnection structures.

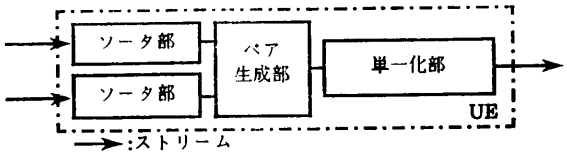


図4 単一化エンジンの構成
Fig. 4 Configuration of UE.

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では，内包的な定義を展開してすべての解を導出するため，RKB内の項関係に対し RBU 演算を繰り返し施すことが要求される．このような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，2属性 (属性名は *head* と *body*) の項関係 *Pr* に格納した場合の検索手順を，**図5**に示す[5]．*Pr* には，各ホーン節のヘッド部を *head* 属性の属性値またボディ部を *body* 属性の属性値として格納する．図5では，Prolog プログラムに与え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，*head* 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 の *body* 属性と *Pr* の *head* 属性間で単一化結合 ($Pr \overset{\bowtie}{\diamond} Tr_0$) 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

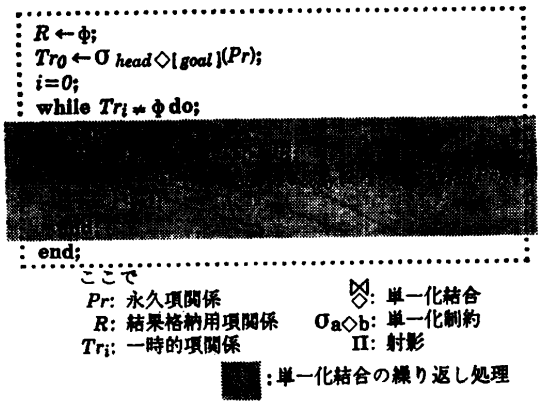


図 5 関係型知識ベースの検索手順
Fig. 5 Retrieval procedure in the relational knowledge base.

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 ($Tr_{i.body}=[\ ]$) となったタップルを集めて求める．

以下では，図5に示すような単一化結合の繰り返しによる検索手順を，キュエリ処理と呼ぶ．また，項関係 Pr は，キュエリ処理の間変更されないため永久項関係と呼び，単一化結合の出力 Tr_i を一時的項関係と呼ぶ．

3.2 単一化結合演算の分割方式

単一化結合演算を複数UEを用いて並列実行させる場合の，演算の分割方式について考察する．

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で盛んに行われている[9),10)]．項関係を対象とする場合でも，変数に代入を行う場合の作用範囲がタップル内に限定されるため，並列処理に関して同様の議論が可能である．

本稿では，2つの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)]について考察する．

ここで，以降で使用する記号をまとめる．まず，項関係を分割したときの個々のタップル集合を，セグメント (segment) と呼ぶ．次に，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．ここで，項関係は MPPM 上に格納することを前提とするため，分割は MPPM のページサイズを単位として行う．

項関係 Q と R を，各セグメントサイズ $|s_Q|$ と $|s_R|$ によりセグメント $q_1,\cdots,q_m$ と $r_1,\cdots,r_n$ に分割した場合，Q と R 間の結合演算は以下のように分割できる．

$$Q\overset{\bowtie}{\diamond}R=\bigcup_{i=1}^{m}\bigcup_{j=1}^{n}q_i\overset{\bowtie}{\diamond}r_j \qquad (1)$$

ここで，$m=|Q|\,//\,|s_Q|$，$n=|R|\,//\,|s_R|$ である．以下では，(1)式の各 $q_i\overset{\bowtie}{\diamond}r_j$ を部分問題，また $Q\overset{\bowtie}{\diamond}R$ から $q_i\overset{\bowtie}{\diamond}r_j$ を生成する処理を問題分割と呼ぶ．

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．ここで入力量を考えるのは，ストリーム処理を実行するUEを用いた場合，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4に示すように，転送速度の等しい入力ポートを2つ持ち，2つの入力データを並列に読み込む．このため，サイズの大きい方のデータの入力時間のみが，UEの処理時間に加算される．以下で入力量とは，サイズの大きい方のデータ入力量を指す．

Q と R 間の単一化結合を(1)式のように分割して実行した場合，$|s_R|\leqq|s_Q|$ とすると，$|s_Q|$ が各部分問題を実行するUEの入力量となる．このため，全部分問題を実行した場合の入力量の合計 I_{Dev} は，

$$\begin{aligned}I_{Dev}&=|s_Q|\times(|Q|\,//\,|s_Q|)\times(|R|\,//\,|s_R|)\\&\geqq|Q|\times(|R|\,//\,|s_R|)\qquad(2)\\&\quad //\text{の定義より}\ |s_Q|\times|Q|\,//\,|s_Q|\geqq|Q|\end{aligned}$$

となる．これに対し，$|R|\leqq|Q|\leqq C_{bf}$ (C_{bf} はUEのバッファサイズ) とし，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 $I=|Q|$ となる．I_{Dev} と I を比較すると $I_{Dev}/I=|R|\,//\,|s_R|$ となり，入力量が分割数に比例して増加する．

以上のように，結合演算の入力となる2つの項関係を分割して，セグメントの全組み合わせを実行するように部分問題を生成すると，入力量は分割数に比例して増加してしまう．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．以下ではこの処理量の増加を，問題分割による分割損と呼ぶ．

問題分割による分割損は，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み合わ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と制御方式による効果を明確にするため，ハッシング等を用いた方式については言及しない．

3.3 キュエリの並列実行制御方式

RKB のキュエリ処理に前節の問題分割方式を適用

し，複数 UE による並列処理を実現する．

図5に示すキュエリ処理では，単一化結合の一方の入力は常に永久項関係 (Pr) であり，もう一方の入力は前段の単一化結合によって作られる一時的項関係 (Tr_i) である．この単一化結合の繰り返しに，前節で述べた問題分割方式を適用するには，前段の単一化結合の結果が出た時点で，永久項関係との間で動的に問題分割を行う制御が必要となる．**図6**にこの制御手順を示す．

図6で，問題分割により生成された各部分問題は，UE への割り当てを待つため，UE 割り当て待ちキュー (UEq) に登録する．ここで，UEq から UE への割り当ては単純なディスパッチ方式による．すなわち，$UE_1, \cdots, UE_n$ に空きができた（割り当てられた部分問題の実行が終了した）時点で，UEq から部分問題を1つ取り出し空き UE へ割り当てる．

問題分割は，各 UE による部分問題の実行が終了した時点で，各 UE の出力結果ごとに独立に行う．ただし，複数の UE からの出力を対象とするため，各 UE の実行結果は，いったん UE 出力キュー (Oq) に集めてから，問題分割への入力とする．問題分割は，Oq に出力結果が有る限り行う．問題分割で生成する部分問題の数は，以下で述べる並列処理方式による．

キュエリ処理の終了は，UEq 上の部分問題と Oq 上の中間結果が無くなった時点とする．また，図5のキュエリ処理では最初に単一化制約を行うが，これは Pr のみをセグメントに分割して部分問題を生成する．

3.4 単一化検索演算の並列処理方式

問題分割として，①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，②部分問題の大きさを可変にして生成する部分問題の数を固定するMP(multiple page at a time) 方式について考察する．

3.4.1 SP 方 式

最初に，問題分割を適用する場合の最も単純な方式として，永久項関係と一時的項関係のセグメントサイズを，MPPM のページサイズに固定する方式について考察する．この方式では，部分問題は常に MPPM の1ページ間の演算になるため，SP 方式と呼ぶ．

SP 方式による問題分割処理は，以下のようになる．

まず，MPPM のページサイズを $|p|$ とし，図6に示す UE_i $(1 \leqq i \leqq n)$ のセグメント s' と s'' に対する出力を $UE_i(s', s'')$ と書く．また，永久項関係 Pr と

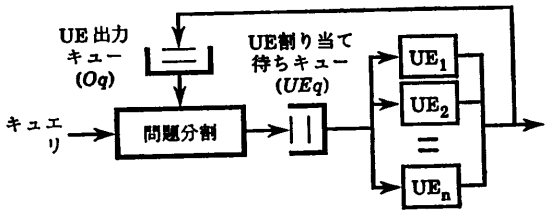


図6 キュエリ処理の並列制御手順
Fig. 6 Control for parallel execution of a query processing.

$UE_i(s', s'')$ を $|p|$ で分割した場合のセグメントを，各々 $t_1, \cdots, t_l$ (ここで，$l = |Pr| \,//\, |p|$) と $u_{i,1}, \cdots, u_{i,m}$ $(m = |UE_i(s', s'')| \,//\, |p|)$ と書く．

SP 方式では，セグメントサイズを固定するから，キュエリ処理の途中で変化しない Pr を，あらかじめ $t_1, \cdots, t_l$ に分割しておくことができる．これにより，問題分割では，各 UE における部分問題の実行が完了した時点でその時の出力である $UE_i(s', s'')$ を分割し，$t_1, \cdots, t_l$ との組み合わせを求めて

$$Pr \otimes UE_i(s', s'') = \bigcup_{j=1}^{l} \bigcup_{k=1}^{m} t_j \otimes u_{i,k} \qquad (3)$$

という部分問題を生成すればよい．

SP 方式によるキュエリの処理時間は，並列実行する UE 台数とページサイズの関係で変動する．すなわち，MPPM のページサイズが小さく UE 台数が少ない場合には，UE 台数に対して過大な部分問題を生成し，分割損により並列処理の効率を低下させる．また，UE 台数が多い場合，ページサイズを大きくすると UE 台数に対して十分な部分問題が生成できなくなる．しかし，部分問題の大きさが一定であるため，複数の UE に対する負荷分散が容易になる．さらに，永久項関係をあらかじめ分割しておくため，問題分割処理を単純化できる．

3.4.2 MP 方 式

SP 方式に対して，問題分割で生成する部分問題数（これを以下で並列度と呼ぶ）を固定する方式を考える．この方式で生成した部分問題は，複数の MPPM ページからなるセグメント間の演算となり，MP 方式と呼ぶ．

MP 方式における，問題分割は以下のようになる．

Pr と $UE_i(s', s'')$ を前節と同じ意味とし，各々を分割するときのセグメントサイズを $|s_p|$ と $|s_o|$ とする．また，分割したセグメントを各々 $t_1, \cdots, t_l$ $(l = |Pr| \,//\, |s_p|)$ と $u_{i,1}, \cdots, u_{i,m}$ $(m = |UE_i(s', s'')| \,//\, |s_o|)$ とする．MP 方式では，問題分割

$$Pr \bowtie UE_i(s', s'') = \bigcup_{j=1}^{l} \bigcup_{k=1}^{m} t_j \bowtie u_{i,k} \qquad (4)$$

において，並列度（これを n とする）を固定するため，

$$m \times l = n$$

となるように $|s_p|$ と $|s_o|$ を調整する．

MP 方式では，UE の出力が完了した時点で，部分問題を実行したときの総入力量を最小にすることを目標として，$|s_p|$ と $|s_o|$ を以下のように決定する．まず，(4)式で分割した部分問題を実行したときの入力量の合計 I_{Dev} を計算すると，

$$\begin{aligned} I_{Dev} &= n \times \max(|s_p|, |s_o|) \\ &\geqq n \times \max(|Pr|/m, |UE_i(s', s'')|/l) \\ &\geqq n \times \{(|Pr|/m + |UE_i(s', s'')|/l)/2\} \\ &\quad \text{ここで，等号は } |Pr|/m = |UE_i(s', s'')|/l \text{ のとき} \\ &= (l|Pr| + m|UE_i(s', s'')|)/2 \\ &\quad \because \quad n = m \times l \\ &\geqq \sqrt{m|Pr| \times l|UE_i(s', s'')|} \qquad (5) \\ &\quad \text{ここで，等号は } |Pr|/m = |UE_i(s', s'')|/l \text{ のとき} \end{aligned}$$

となる．計算過程から，I_{Dev} は $|Pr|/m = |UE_i(s', s'')|/l$ のとき最小になる．MP 方式では，$|Pr| \times |UE_i(s', s'')|$ を n 個に等分割しその平方根を部分問題のセグメントサイズとする．すなわち，$|s_p| = |s_o| = \sqrt{(|Pr| \times |UE_i(s', s'')|)/n}$ とする．ただし，このとき MPPM のアクセス単位はページであるため，$|s_p|$ と $|s_o|$ は MPPM のページサイズで量子化する．また，UE のバッファサイズ C_{bf} に対し，$|s_p| \leqq C_{bf}$ かつ $|s_o| \leqq C_{bf}$ でなければならない．しかし，$|Pr| \times |UE_i(s', s'')| \geqq (n \times C_{bf}^2)$ の場合（特に，$|Pr| \geqq n \times C_{bf}$ の場合）は，$|s_p| = C_{bf}$ として n 個以上の部分問題を生成する．この時，$|UE_i(s', s'')| \geqq C_{bf}$ であれば $|s_o| = C_{bf}$ とし残り（$|UE_i(s', s'')| - C_{bf}$）は次の問題分割にまわす，また $|UE_i(s', s'')| < C_{bf}$ であれば $|s_o| = |UE_i(s', s'')|$ とする．

MP 方式では，セグメントサイズを MPPM のページサイズより小さくしないため，部分問題の数は常に SP 方式以下になる．このため，並列実行する UE 台数が同じ場合は，分割損が小さくなるだけ MP 方式の方が全体の処理量を小さくできる．SP 方式に対して，問題分割にセグメントサイズを計算する時間が加算されるが，並列度に比例する時間とはならない．

4. 処理性能

以下では，SP 方式と MP 方式を図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，すべてシミュレーション[15]による．

シミュレーションでは，図6に示す制御手順をモデルとし，問題分割に SP 方式と MP 方式を適用した場合の，キュエリ処理の処理時間（ターンアラウンドタイム）を測定した．特に，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．実行したキュエリは，親子関係から先祖を検索する問題であり，64 KB の永久項関係を対象とする．

UE については，バッファサイズを最大 64 KB，また処理速度は 20 MB/s と仮定した[14),16)]．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．また，図3に示したように，MPPM は複数ポートに対して競合なくデータの転送が可能であるから，MPPM アクセス時の衝突は仮定せずシミュレーションを行った．

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，UE 台数を固定してページサイズごとに求めた結果を図7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よって決まる．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問題分割による分割損が小さくなり，処理時間が短くなる．一方，ページサイズ

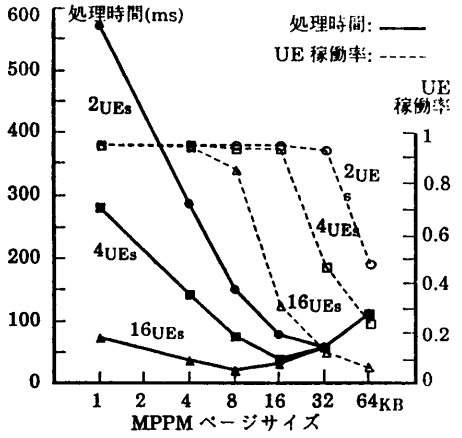


図7 SP 方式における処理時間と UE の稼働率
Fig. 7 Elapsed time and UE utilization in SP method.

が大き過ぎると UE 台数分の部分問題が生成されず，UE の稼働率が低下して処理時間を悪くする．しかし，UE 台数が多い場合，問題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，処理時間の変化は小さくなる．

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．すなわち，ページサイズが小さくて UE 台数が少ない場合は，UE 台数に対して十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 台数に対して十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．また，各部分問題の大きさが一定であるため，部分問題数が十分に生成できるページサイズの範囲では，安定した稼働率が得られている．

以上のように，セグメントサイズを固定して問題分割を行うと，UE 台数や項関係の大きさの変動に対して，常に最適な処理時間を得ることができなくなる．

4.2 MP 方式の性能特性

MP 方式による処理時間と稼働率のグラフを図 8 に示す．ここでは，問題分割の並列度を UE 台数と等しくして測定した．

MP 方式では，部分問題の数が UE 台数によって決まる．このため，UE 台数分の部分問題が生成できるページサイズの範囲では，処理時間はページサイズに依存せず，安定した処理時間が得られる．特に SP 方式に対して，ページサイズが小さい場合の処理時間が，短くなっている．これは，UE 台数が小さい場合に，SP 方式のように UE 台数に対して過大な部分問題を生成することがなくなり，問題分割による分割損を低く押さえることができたためである．また，ページサイズが大きくなると，処理時間は SP 方式の処理時間に近づく．これは，ページサイズが大きくなると，ページサイズをセグメントとして問題分割を行うようになり，SP 方式と同じ問題分割になるためである．

稼働率については，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 方式と同じ理由による．また，SP 方式に対して，稼働率のばらつきが見られるが，これは部分問題の大きさが問題分割で対象とした UE の出力量により，変動するためである．

4.3 UE バッファサイズの影響

図 8 と同じ条件で，UE のバッファサイズ (C_{bf}) を変えて計測した結果を，図 9 に示す．

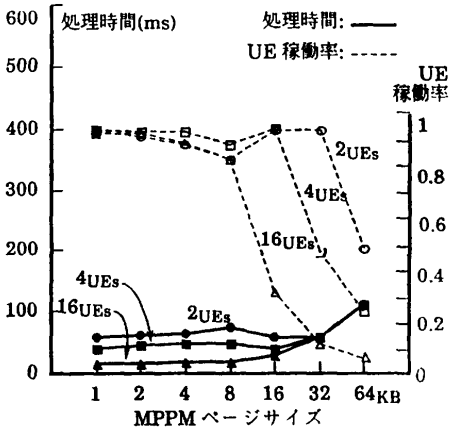


図 8 MP 方式における処理時間と UE の稼働率
Fig. 8 Elapsed time and UE utilization in MP method.

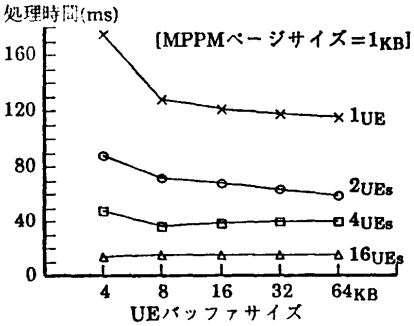


図 9 UE バッファサイズの影響
Fig. 9 Effect of C_{bf}.

並列度が増加するほど，処理速度に対する C_{bf} の影響が少なくなっている．これは，問題を常に UE 台数で分割しているために，並列度が大きくなるとセグメントサイズが C_{bf} より小さくなり，問題分割が C_{bf} の影響を受けなくなることによる．これに対して，並列度が小さい場合は，問題を並列度で分割すると部分問題のセグメントサイズが C_{bf} より大きくなってしまう．このため，C_{bf} をセグメントサイズとして問題を分割するようになり，C_{bf} が大きくなると部分問題の数が減少して分割損が少なくなり，処理時間が短くなる．

5. 議　　論

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，並列処理の効果がほとんど無くなってしまう．これに対し

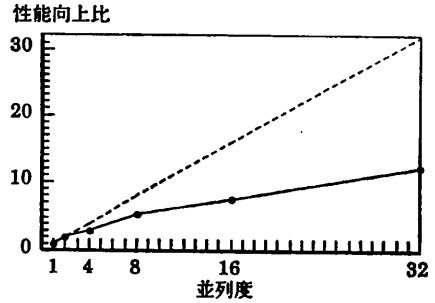


図 10 並列処理効果
Fig. 10 Effect of parallel processing.

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，問題分割による分割損を完全に除くまでは至っていない．例えば，**図 10** を見ると，制御オーバヘッドを入れていないのに，性能向上比は並列度に比例していない．これは，MP 方式では並列度が大きくなると問題の分割が細かくなり，並列度の増加に従って問題分割による分割損が増加し，キュエリ処理の処理効率を低下させるためである．

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．本稿で述べたシミュレーションでは，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，全く出力の無いものもある．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．RKB のキュエリ処理でも，文献 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み合わせの省略が可能と思われる．

6. むすび

以上，RKB に対するキュエリ処理を効率的に並列処理する制御方式について考察した．検索演算を並列実行するときの方式として，SP 方式と MP 方式の 2 方式を考え，シミュレーションによる評価を行った．

RKB のキュエリ処理では，項関係間の結合演算が繰り返し実行され，キュエリ処理の効率化にはこの結合演算の効率化が不可欠となる．結合演算を並列処理により高速に処理するため，項関係を分割して並列実行可能な部分問題を生成すると，入力量が増加し処理効果を低下させる．MP 方式では，常に並列度分の問題分割しか行わないことで，生成する部分問題の数を低く押さえ比較的安定した処理時間を得ることができるようになった．

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．

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いたします．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*, 3 (2), OHMSHA, Tokyo (June 1985).
2) Ito, N. et al.: The Dataflow-based Parallel Inference Machine to Support to Basic Languages in KL1, *Proc. IFIP TC-10 Working Conf. of Fifth Generation Computer Architecture*, UMIST (Manchester) (July 1985).
3) Moto-oka, T. et al.: The Architecture of a Parallel Inference Engine—PIE—, *FGCS '84*, pp. 479-488, ICOT (1984).
4) Kakuta, T. et al.: The Design and Implementation of Relational Database Machine Delta, *Proc. Int. Workshop on Database Machine '85*, pp. 13-34 (March 1985).
5) Yokota, H. et al.: A Model and an Architecture for a Relational Knowledge Base, *Proc. 13th Ann. Int. Symp. Computer Architecture*, pp. 2-9 (June 1986).
6) Tanaka, Y.: A Multiport Page-Memory Architecture and A Multiport Disk-Cache System, *New Generation Computing*, 2, pp. 241-260, OHMSHA, Tokyo (February 1984).
7) Morita, Y. et al.: Retrieval-By-Unification Operation on a Relational Knowledge Base, *Proc. 12th Int. Conf. VLDB*, pp. 52-59 (August 1986).
8) Itoh, H. et al.: Parallel Control Technique for Dedicated Relational Database Engines, *Proc. 3rd Int. Conf. on Data Engineering*, pp. 208-215 (1987).
9) Valduriez, P.: Semi-Join Algorithms for Multiprocessor Systems, *ACM Trans. Database Syst.*, Vol. 9, No. 1, pp. 133-161 (1984).
10) Boral, H. et al.: Processor Allocation Strategies for Multiprocessor Database Machines, *ACM Trans. Database Syst.*, Vol. 6, No. 2, pp. 227-254 (June 1981).
11) Boral, H. et al.: Design Consideration for Data-flow Database Machines, *Proc. ACM-*

SIGMOD Int. Conf. Management of Data, pp. 95-104 (1980).
12) Kitsuregawa, M.: Architecture and Performance of Relational Algebra Machine GRACE, *Proc. Int. Conf. Parallel Processing*, pp. 87-96 (1984).
13) DeWitt, D.: Multiprocessor Hash-Based Join Algorithms, *Proc. Int. Conf. VLDB*, pp. 151-164 (1985).
14) 伊藤ほか：大規模知識ベースマシンの開発(1)-(4)，第33回情報処理学会全国大会論文集，3B-1～3B-4 (1986).
15) 柴山ほか：大規模知識ベースマシン実験機の開発(2)～(3)，第34回情報処理学会全国大会論文集，3P-2～3P-3 (1987).
16) 森田ほか：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，情報処理学会研究会報告，DBS 57-4 (1987).
17) 森田ほか：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，第33回情報処理学会全国大会論文集，6L-8 (1986).
18) 大森ほか：推論機能と関係データベースの融合方式，信学技報，AI 86-21 (1986).
19) Kiyoki, Y.: A Relational Database Machine Based on Functional Programming Concepts, *Proc. FJCC*, pp. 969-978 (1986).
20) 喜連川ほか：データベースマシン，情報処理，Vol. 28, No. 1, pp. 56-67 (1987).

(昭和62年9月21日受付)
(昭和63年3月9日採録)

物井 秀俊（正会員）

昭和30年生．昭和53年茨城大学工学部情報工学科卒業．昭和55年東北大学大学院工学研究科修士課程修了．同年(株)日立製作所入社．同年より同社ソフトウェア工場にて大型計算機のオペレーティングシステムの開発に従事．昭和60年(財)新世代コンピュータ技術開発機構に出向．現在同機構研究所において，知識ベースマシンの研究・開発に従事．

森田 幸伯（正会員）

昭和33年生．昭和56年東京理科大学理学部応用数学科卒業．昭和58年東京理科大学大学院理学研究科修士課程修了．同年沖電気工業(株)入社．同年より同社総合システム研究所にて，データベースマシン，および，ネットワークシステムの研究・開発に従事．昭和60年(財)新世代コンピュータ技術開発機構に出向，現在同機構研究所において，知識ベースマシンの研究・開発に従事．応用統計学会会員．

伊藤 英則（正会員）

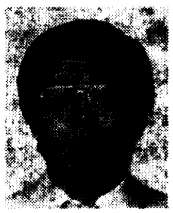
昭和21年生．昭和44年3月福井大学工学部卒業．昭和49年3月名古屋大学大学院工学系研究科博士課程電気及電子専攻修了，工学博士．昭和49年4月，NTT通信研究所入社．現在，(財)新世代コンピュータ技術開発機構に勤務，研究室長．これまでに，オートマトンと数理言語理論の研究と大型計算機の研究開発に従事．現在，知識情報処理システムの研究開発に従事．電子情報通信学会，人工知能学会各会員．

岩田 和秀（正会員）

昭和18年生．昭和43年名古屋大学工学部電気工学科卒業．昭和48年同大学院博士課程修了．同年東京芝浦電気(株)（現(株)東芝）入社，総合研究所に勤務．電力変換装置，産業用ロボット，専用プロセッサなどの研究・開発に従事．昭和62年(財)新世代コンピュータ技術開発機構に出向．現在，同機構研究計画部にて海外交流を担当．電気学会会員．

酒井 浩（正会員）

昭和30年生．昭和53年東京大学工学部計数工学科卒業．昭和55年同大学院修士課程修了．同年東京芝浦電気(株)（現(株)東芝）入社．総合研究所情報システム研究所において，知識ベースマシンの研究・開発に従事．人工知能学会会員．



柴山 茂樹（正会員）

昭和28年生．昭和50年東京大学工学部電子工学科卒業．同年東京芝浦電気(株)（現(株)東芝）入社．総合研究所情報システム研究所において，アレイプロセッサ，CT用画像再構成プロセッサ，データベースマシン，知識ベースマシンの研究・開発に従事．この間，昭和57年～60年(財)新世代コンピュータ技術開発機構に出向．電子情報通信学会会員．