計算機アーキテクチャ 79－14
（１９８９．１１．２１）

リング結合型並列計算機のシステムアーキテクチャ

中條 拓伯* 和田 耕一** 金田 悠紀夫*

＊ 神戸大学 工学部 システム工学科
＊＊ 筑波大学 電子情報工学系

リング結合型並列計算機は疎結合でありながら，ネットワーク仮想共有メモリなど，密結合マルチプロセッサのもつ利点をも有効に活かすためのマシンであり，１０２４台までの拡張を目指して現在開発中である．ここではシステム全体のアーキテクチャと処理結果の回収のためのホストコンピュータと各要素プロセッサ（ＰＥ）との通信形態と割込みについて述べ，ネットワークの性能を低下させないためのフロー制御について説明する．さらに，Ｂ－ｔｒｅｅをＰＥ上に構築し，並列に検索，挿入，削除する方法について解説する．

System Architecture of Ring-Connected Parallel Computer

Hironori Nakajo* Koichi Wada** Yukio Kaneda*

* Department of Systems Engineering,
Faculty of Engineering, Kobe University
Rokkodai-cho, nada-ku, Kobe, 657 Japan,

** Institute of Information Science and
Electronics, University of Tsukuba
1-1-1, Tennodai, Sakuramura, Niharigun,
Ibarakiken, 305 Japan

Though our Ring-Connected Parallel Computer is a loosely-coupled multi-processor, equipped with network virtual shared memory, it has characteristics of a tightly-coupled multi-processor. This parallel machine is now being developed with 32 processors, but, in the future, we aim the expansion to 1024 processors.

We describe the system overall configuration, communication method between Host computer and each processing element(PE)s to collect answers processed in each PE, interrupt mechanism, and flow control not to deteriorate network performance.

Moreover, we explain the parallel search, insertion, and deletion using B-tree constructed in each PE.

1．はじめに

リング結合型並列計算機はデータベース，知識ベースの並列処理，さらにネットワーク上での仮想共有メモリなどに関する研究を行うための実験システムとして現在，開発が進んでいる．

ここでは，本システムの構成と割込み機構について述べ，さらにネットワーク内の輻輳を回避するためのフロー制御について説明する．

そして，本システムにおいてB－ｔｒｅｅによる検索，挿入，削除を並列に行う方式を提案し，その可能性について考える．

2．システムアーキテクチャ

図１にリング結合型並列計算機のシステム全体の構成を示す．隣接する要素プロセッサ（ＰＥ）はストリーム・コントローラ（ＳＣ）とよぶ通信専用ハードウェアによってリング状に結合されている．ＰＥ間はＳＣにより，高速にパケット通信が行われ，負荷分散やいろいろなメッセージ通信に用いられる．さらに，ホストコンピュータから各ＰＥはブロードキャスト・バスで接続されており，ホストからＰＥへ同一のデータが同時に転送（放送）される．このとき，すべてのＰＥに対して，またはいくつかのＰＥに対して（受信者選択型マルチキャスト），さらに個々のＰＥに対してそれぞれ選択して転送できる[1][2]．

表１に各ＰＥのハードウェア仕様を示す．

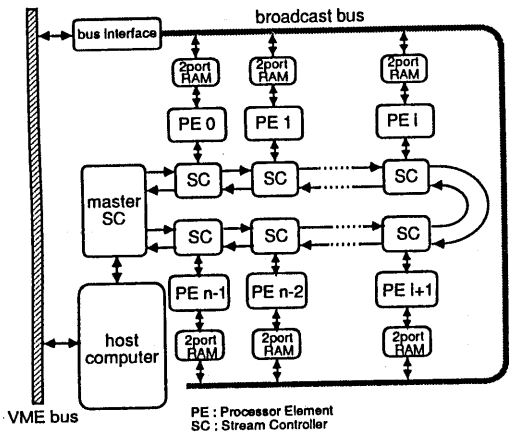

図１　リング結合型並列計算機のシステム構成

規模：	最大　１０２４プロセッサ
接続形態：	リング＋ブロードキャスト
ＣＰＵ：	ＭＣ６８０２０（１６ＭＨｚ）
ローカルメモリ：	ＲＯＭ　１２８Ｋｂｙｔｅ
	ＤＲＡＭ　２Ｍｂｙｔｅ
放送メモリ	：ＳＲＡＭ　５１２Ｋｂｙｔｅ

表１　各ＰＥのハードウェア仕様

2．1　システムのメモリマップ

図２にホストコンピュータと各ＰＥのメモリマップを示す．各ＰＥは放送領域として５１２ＫバイトのＳＲＡＭを実装しているが，この領域は，図２に示すようにホストのメモリーマップ上にもマッピングされており，ホストは１０２４台のＰＥから構成した場合，最大５１２Ｍバイトの実メモリを実装することになる．どのＰＥに対して転送するか，すなわち，どのようにマルチキャストを行うかは，４Ｇという広大なメモリ空間を有するＭＣ６８０２０のアドレスバスを有効に用いて決定される[1]．

各ＰＥは２ＭバイトのＤＲＡＭを実装しているが，その一部をネットワーク仮想メモリとして用い，他のＰＥと共有され，仮想的に個々のＰＥは最大５１２Ｍバイトのネットワーク仮想共有メモリを有することになる．これについては後述する．

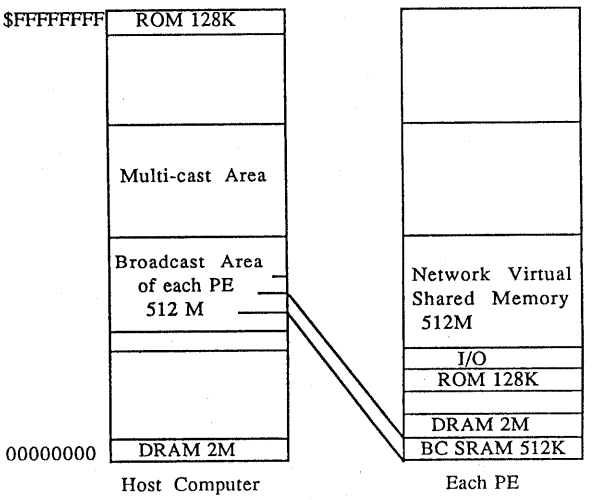

図２　メモリマップ

2．2　割込み

マルチプロセッサシステムにおいて，ホストコンピュータが各ＰＥへの処理開始の合図や，各ＰＥから処理の終了の報告を行うときに割込みが有効に用いられる．ここでは，ホストコンピュータから各ＰＥへ，また各ＰＥからホストへの２種類の割込みについて説明し，その有効性について述べる．

（1）ホストコンピュータから各ＰＥへの割込み

データベース，知識ベースなどの大量のデータを分割し，各ＰＥに分配する場合を考える．ＰＥ側でローディングの終了を知る方法としてＥｎｄ　ｏｆ　Ｄａｔａ（ＥＯＤ）を表す特殊なデータを用いる．ここで，ＰＥが他に行うべき処理がなければ，ＥＯＤをポーリングすることによって処理の開始時を知ることができるが，他の処理を行っている場合は割込みが有効に用いられる．

具体的には，前に述べたマルチキャストを用いて，ホストから個々のＰＥ，または複数のＰＥに対して割込みをかけることができる．つまり，ホストが各ＰＥの放送領域の最終アドレスにデータを書き込んだ場合に割込み信号を発するが，そのときのアドレスによってどのＰＥに対して割込みをかけるかを容易に決定できる．

割込みをかけられたＰＥは，そのときに優先順位のより高い割込みがかかっていなければ，ホストからの割込みに対するアクナリッジ信号をオープンコレクタバスにＨ（Ｈｉｇｈ）として出力する．ＰＥがたとえ一つでもアクナリッジ信号をＬ（Ｌｏｗ）で返すと割込みは受け付けられず，アクナリッジがＨになるまで待たされることになる．

（2）各ＰＥからホストコンピュータへの割込み

それぞれのＰＥにおいて処理した結果の回収について考える．各ＰＥの放送領域はホストコンピュータのメモリマップ上にマッピングされているので，ＰＥはその領域に処理結果を書き込み，ホストが読みにいくことで容易に結果を回収できる．

例えば，データベースをＰＥに分配し，検索データを放送して，検索を行う場合を考える．各ＰＥの放送領域にステータスフラグを用意し，ＰＥが処理を終了したかどうか，解を持つかどうかなどの情報を蓄える．このとき，

ａ．　多数のＰＥが解を持つ場合

ホストは各ＰＥのステータスフラグを順に読み込み，処理が終了していれば，結果を読み込んで回収する．ＰＥがまだ実行中であれば，そのＰＥのＩＤをキュー（process queue）に書き込み，次のＰＥのステータスフラグを読み込む．以上の操作をすべてのＰＥに対して行い，１通りの処理が終了すれば，キューに格納されたＰＥに対して，再びステータスフラグを読み込み，キューが空になるまで続ける．以上の操作を図３に示す．

```
for (i = 0; i < n; i++) {
      if (status_flag(i) != finish)
            process_enqueue(i);
      else
            collect_result(i);
}

do {
      id = process_dequeue();
      if (status_flag(id) != finish)
            process_enqueue(id);
      else
            collect_result(id);
} while (process_queue == empty);
```

図３　複数ＰＥからの結果回収方法

ｂ．　少数のＰＥが解を持つ場合

この場合，ホストが各ＰＥのステータスフラグを順次チェックするのは効率が悪い．それよりＰＥ側から解を持っていることをホストに知らせる方が有効と考えられる．そこで，ＰＥからホストへの割込みが用いられる．

ＶＭＥバスなど，単一バスに複数のプロセッサが接続されているようなマルチプロセッサシステムでは，個々のボードからの割込みはディージ・チェーンによって排他制御され，どのボードからの割込みかを判断するには，各ボードから送出されるベクタを用いる．

しかし，ＭＣ６８０２０の場合，ベクタ・テーブルは２５６個に限られており，さらにＭＰＵ自身が６４個のベクタを，アドレス・エラーやゼロ除算などのために定義している．したがって，ベクタによってインタラプタを判断するには，１９２台までが限度であり，１０２４台までの拡張を考えている本システムでは使用は困難である．さらに，ディージ・チェーンを用いた場合，その伝搬時間も見過ごすことのできない問題となる．

そこで，本システムでは割込みのための割込みＩＤバスとよばれる特別なバスを設け，他のＰＥからの割込みの競合に対して排他制御を行っている．ホストは１０２４台のＰＥの中で，どのＰＥからの割込み要求かを容易に判断でき，それぞれの要求に対して応答する．ＰＥからの割込み発生のためのハードウェア・ブロック図を図４に示す．

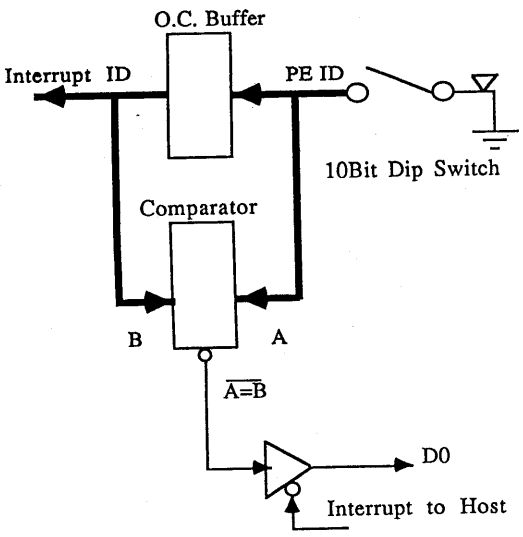

図４　ＰＥからの割込みのブロック図

ＰＥからホストへの割込みシーケンスについて説明する．

①　ＰＥが割込みをかける場合，あるポートを読みにいく．

②　そのポートのアドレスをデコードして，自身のＩＤをインターラプト・ＩＤ（ＩＮＴ・ＩＤ）バスに乗せる．

③　ＩＮＴ・ＩＤバスはオープンコレクタ出力で他のＰＥのＩＮＴ・ＩＤバスと接続されており，他に割込みをかけているＰＥがなければ，自身のＩＤとＩＮＴ・ＩＤとが一致し，その結果をデータバスの最下位ビット（Ｄ０）に乗せる．①でポートを読みにいくと言うのは，実はこのＤ０を読みにいくことであり，もし０ならば他に競合することがなく，ホストに割込みを受け入れられたと判断でき，１ならば他のＰＥと競合し，割込みが受け入れられなかったと判断できる．①から③までは，ＭＣ６８０２０の単なるリードの１サイクルであり，ノー・ウェイトもしくは１・ウェイトで動作する．

④　ホストは割込みを受けつけた後，このＩＮＴ・ＩＤを読みにいくことで，どのＰＥからの割込みを容易に判断できる．ＩＮＴ・ＩＤは１０ビット用意してあるので，ホストは１０２４台までのＰＥからの割込みを判断することができる．

⑤　クリティカルなタイミングで，２つ以上のＰＥが同時に割込みをかけた場合，ホストへの割込みはかからない．

ＰＥからホストへの割込み操作を図５に示す．

```
do {
    check = interrupt_to_host();
    if (LSB(check) == 1)
        do_another();
} while (LSB(check) == 0);
```

図５　ＰＥからホストへの割込み

3．フロー制御とデッドロックの回避

複数のノードからなるネットワーク上の問題として輻輳（congestion）の発生がある．これはメッセージやデータが一箇所，または数カ所に集中するために起こる．このため，メッセージの伝達速度は急速に低下し，最終的にはデッドロックに陥る恐れがある．

一般的なネットワーク上では次の２種類のデッドロックが起こり得る[6]．

a．　プロトコル・デッドロック

受信確認応答や送信許可など，ある種の制御信号のような論理的な資源が原因となって発生する．

b．　バッファ・デッドロック

メッセージ・バッファなどの物理的な資源が原因となって発生する．

本システムでは，ネットワークの性格上，バッファ・デッドロックが起こる可能性が高いと考えられる．ネットワークにおいて輻輳を防ぐためのフロー制御には，ネットワーク内でただ１つのノードが制御する集中型フロー制御と，各ノードで制御する分散型フロー制御がある．本システムではブロードキャスト・バスを用いて集中的に管理することも可能であるが，ホストの負荷を考慮して，分散型を選択する．一般的なフロー制御には次の３つの方法がある[6]．

・適応ルーティング
・エンドツーエンド・フロー制御
・イザリズミック・フロー制御

ルーティングを不要にするためにリング構造を採用した本システムでは，適応ルーティングは無関係である．エンドツーエンド・フロー制御は受信先の状態にもとづいて送信側がメッセージ・トラフィックを発生させる割合を規制するものであるが，本システムでは，受信側の状態を知るようなメッセージのやりとりを行うようなことは考えていない．イザリズミック・フロー制御はネットワーク内のメッセージ総数を一定値に制限することで輻輳を制御するものである[6]．

本システムでは，このイザリズミック・フロー制御を採用する．具体的には，ネットワーク内に一定数のパーミット（入場券）が発行され，このパーミットを得たパケットだけがネットワーク内で転送される．このため，ネットワーク内のパケット数はこのパーミット数以下に保たれ，地域的に多少の輻輳が生じるが，ネットワーク全体に大きな影響を及ぼすことはない．パーミットを得てネットワーク内に入ったパケット，またはすでに伝送されているパケットを内部パケットとよび，それ以外を外部パケットとよぶ．外部パケットが伝送されるためには，そのパケットを持つノードは他のノードからパーミットが配送されるのを待たなければならない．内部パケットには次の２種類のパケットが存在する．

（1）データパケット

通常のメッセージあるいは制御情報を含むパケットで，通常１個のパーミットを有し，転送先のノードでそのパーミットを提供する．本システムでは，データパケットをさらにエンドツーエンド（end-to-end）・データパケットとブロードキャスト（broadcast）・データパケットの２つに分類する．前者はＰＥ間における１対１通信に用いられ，後者は１対多通信に用いられる．このブロードキャスト・データパケットを用いることで，すべてのＰＥは他のＰＥへ放送を行うことができる．

（2）空パケット

このパケットはパーミットの配送のみに用いられ，ある最大数以下のパーミットを一時に運ぶ．ノードにおいてパーミットがあふれた場合，余分のパーミットは，この空パケットを用いることで他のノードに配送される．空パケットはメッセージを含んでいないためネットワーク上のオーバヘッドとなる．

このようにネットワーク内のパーミット数を一定値以下にすることで，ネットワーク全体のパケット数を制限できる．本システムでの実際の制御は以下の手順で行う．

a． 空パケットを受信する場合

① パーミット・バッファがフルならば，そのパーミットをエンドツーエンド空パケットとして，ネットワークの隣接ＰＥに転送する．（図６－１）

② パーミット・バッファがフルでないならば，その空パケットを取り込み，パーミット・バッファにパーミットを追加する．（図６－２）

b． エンドツーエンド・データパケットを受信する場合

① パーミット・バッファがフルならば，データパケットを取り込んだ後，空パケットをネットワーク上に発信する．（図７－１）

② パーミット・バッファがフルでないならば，データパケットを取り込んだ後，パーミット・バッファにパーミットを１つ追加する．（図７－２）

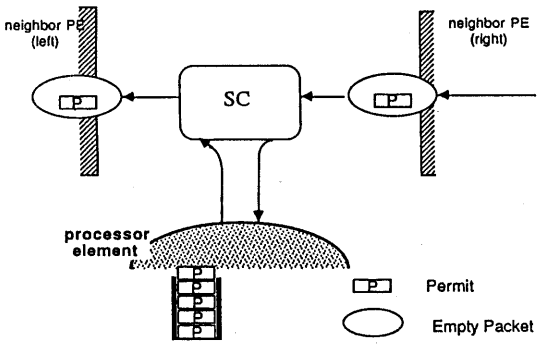

図６－１　空パケットの受信
（パーミット・バッファがフル）

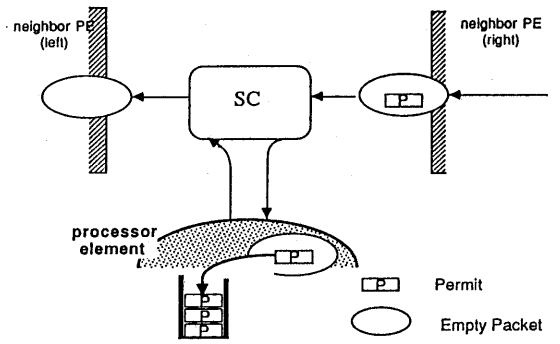

図６－１　空パケットの受信
（パーミット・バッファがフルでない）

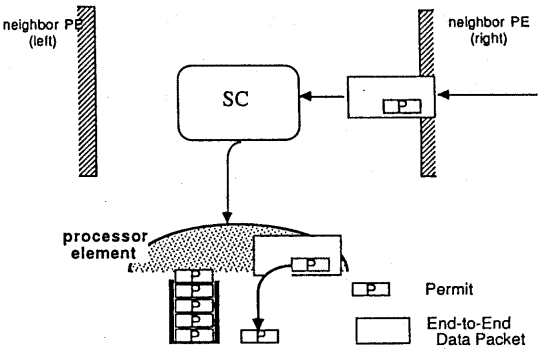

図７－１　エンドツーエンド・データパケット
の受信（パーミット・バッファがフル）

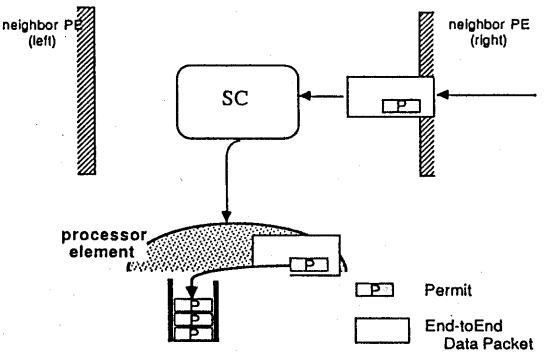

図７－１　エンドツーエンド・データパケット
の受信（パーミット・バッファがフルでない）

c． ブロードキャスト・データパケットを受信する場合（図８）

ブロードキャスト・データパケットはネットワークを巡回し，それぞれのＰＥに取り込まれ，一回りした後に発信ノードによって回収される．このときブロードキャスト・データパケットはエンドツーエンド・データパケットと同様にパーミットを含むが，パーミットの配送をそれぞれのＰＥに対して行えば，パーミット数が著しく増加する．そこで，ブロードキャスト・データパケットはパーミットの配送は行わないことにする．すなわち，各ＰＥはブロードキャスト・データを受け取るが，パーミットは受け取らない．

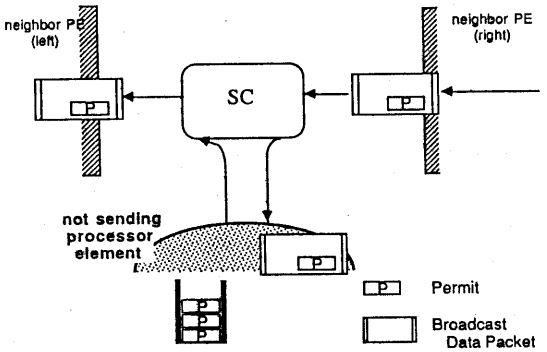

図８　ブロードキャスト・データパケットの
受信

d．　エンドツーエンド・データパケットの送信

（1）パーミット・バッファが空でないならば，エンドツーエンド・データパケットを送信した後，パーミット・バッファからパーミットを１つ削除する．（図９－１）

（2）パーミット・バッファが空ならば，空パケットが隣接ＰＥから転送されパーミットが配送されてくるまで送信を待つ．（図９－２）

e．　ブロードキャスト・データパケットの送信

（1）パーミット・バッファが空でないなら，ブロードキャスト・データパケットを送信した後，パーミット・バッファからパーミットを１つ削除する．

（2）パーミット・バッファが空ならば，空パケットが転送されてくるまで送信を待つ．

ネットワーク上の総パーミット数についてであるが，パーミットが永遠に得られないという飢餓状態を回避するためにネットワーク上には，絶えずいくつかの空パケットを保持する必要がある．パーミット数を多く取れば，パーミット・バッファが空となるＰＥは少なくなるが，空パケットの送受信のためのオーバヘッドが大きくなる．パーミット数を少なくすると，パーミットを持たないＰＥが増加し，データパケットの転送効率は低下する．したがって，転送スピードを考慮して最適なパーミットの分配を行わなければならない．

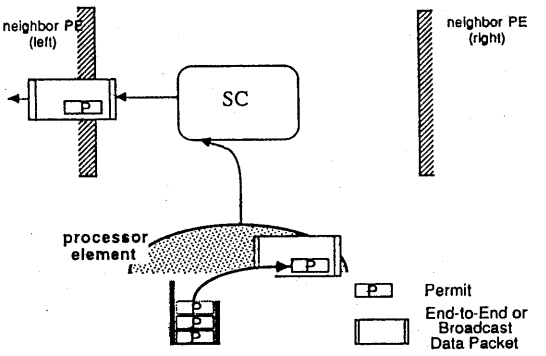

図９－１　エンドツーエンド・データパケットの
送信（パーミット・バッファが空でない）

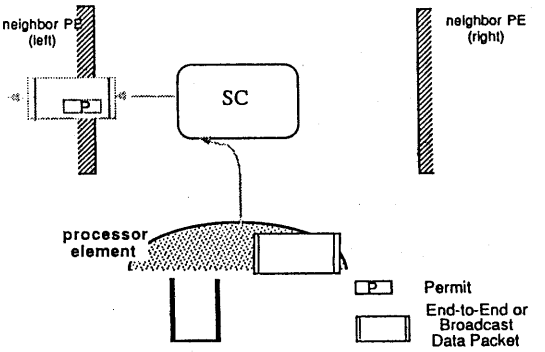

図９－２　エンドツーエンド・データパケットの
送信（パーミット・バッファが空）

４．ネットワーク仮想共有メモリ

本システムでは，高速なパケット通信を用いて，各ＰＥのメモリの一部を他のＰＥと共有することで，仮想的に大容量のメモリを有するというネットワーク仮想共有メモリの実現を構想している．共有メモリの有効性については，比較的少数のＰＥからなる商用機の成功が示す通りであるが，ＰＥ数の増加に対しては悲観的である．そこで，ネットワークを用いたシステムによる仮想共有メモリが今後有利になってくると考えられる．図２のメモリマップに示したように，各ＰＥは５１２Ｍバイトの仮想共有メモリを持つ．現在は，データの無矛盾化プロトコルについて詳細を検討中である．

5．Ｂ－ｔｒｅｅの並列実行方式について

データベースに蓄積されるレコードの数が増加するに従って，その中から特定のデータを取り出すことが困難になってくる．あるキーを与えて，それと一致するキーの値を持つレコードをファイルの中から検索する方法の代表的なものに，ハッシングとインデックスがある．データを検索する場合，ハッシュを用いればO(1)で検索できるが，検索するキーをある範囲で指定した場合にはインデックスの方が有効である．また，データの追加に関して，ハッシングでは追加データが増加するとハッシュテーブルを再編成する必要が生じ，その処理とテーブルに消費するメモリ容量が問題になってくる．ここでは，インデックスの中でも特にＢ－ｔｒｅｅについて，本システムでの並列実行の可能性について述べる．

まず，ホストコンピュータの２次記憶装置にあるデータベースを各ＰＥへ分配する．この場合に次に示す２通りの方法がある．分配を行った後，各ＰＥ内でＢ－ｔｒｅｅの構築を行う．

（１）キーの値において，どのＰＥがどの範囲のデータを処理するかを決定しておき，ブロードキャスト・バスを用いてデータを分配する．

（２）ホストはデータベースを等量に分割し，それぞれを各ＰＥに分配する．

（１）の分配はブロードキャスト・バスを用いれば比較的容易に実現できる．すなわち，個々のＰＥの放送領域がホストのメモリマップ上にマッピングされているので，キーの値から送るべき領域の計算（アドレス変換）を行うのに算術計算だけでよく，複雑な条件処理を行う必要はない．ホストがあるデータを検索，挿入，削除を行うときに，そのデータのキーの値から，どのＰＥ内のｔｒｅｅにあるかを判断できるので，多数のデータを検索する場合，データの検索を行うＰＥに随時転送し，並列に処理を進めることが可能である．ただし，キーの値によって，ＰＥ間でデータ数のばらつきが生じる．この場合，隣接ＰＥ間でデータ転送を行い，データ数がなるべく等しくなるようにする必要がある．

（２）の場合は，（１）よりもデータの分配が高速に行えるが，データの検索は，検索データを放送することで行える．データの挿入を行う場合も，すべてのＰＥにそのデータを放送し，どのｔｒｅｅにも含まれないことがわかれば，データ数の少ないＰＥ内のｔｒｅｅに挿入すればよい．各ＰＥの持つデータ数は，ほぼ等量に保てるので負荷分散の点では（１）より優れていると思われる．

6．おわりに

現在は３２台からなるプロトタイプを製作中である．Ｂ－ｔｒｅｅの並列処理に関する問題は既存の密結合マシンにおいて評価をおこなっている．今後は，ネットワーク仮想共有メモリについても，その詳細を明確にしていく予定である．

［参考文献］

[1]中條，和田，金田，他：”論理型並列データベースマシンに関する研究”，信学技報，CPSY88-29,(1988)

[2]中條，和田，金田，他：”並列知識ベースマシンのアーキテクチャ”，第３７回情報処理学会全国大会，3N-2,(1988)

[3]中條，和田，金田：”並列知識ベースマシンのハードウェア構成”，並列処理シンポジウムJSPP'89，PP.147-154(1989)

[4]富田：並列計算機構成論，昭晃堂(1986)

[5]Hsiao,D.K.:"The Impact of the Interconnecting Network on Parallel Database Computers", Database Machines and Knowledge Base Machines, Kluwer Academic Publishers(1988)

[6]Ahuja,V.A.:"Design and Analysis of Computer Communication Networks",McGRAW-HILL(1982)