マルチマイクロプロセッサシステム HARPS

Multimicroprocessor System HARPS

データベースマシン および その他の応用

Emulation of database machine and Other Applications

田中 譲・田畑 隆司・津田 孝夫
Yuzuru TANAKA, Takashi TABATA, Takao TSUDA
北海道大学・工学部
Hokkaido University

A new approach to relational data base machine is shown as an application of HARPS. HARPS is used to emulate this data base machine called OPREC (Orthogonal Processor for Relational Computation). This system is based on two important ideas, i.e., 1) Parallel access to files, and 2) Parallel processing. This two kinds of parallelism are both applied to domains of a relation in relational data base. Together with the outline of the OPREC's architecture, various problems on data base machines and their design theory are described.

[1] 序論

北海道大学情報工学専攻では，日本ミニコンピュータ(株)と共同開発になるマルチ・マイクロプロセッサ・システムHARPS[1](Hokkaido University Array Processor System)を現在研究開発中であり，汎用並列計算機の開発を目指したMOSES[1]の他，種々の応用システムの研究開発も開始されつつある。主なものには，DDA(Digital Differential Analyzer)，言語プロセッサの並列処理化，神経回路網のディジタル・シミュレータ，FFT等の特殊計算用高速計算機，データベース・マシン等の研究がある。この報告では，著者等が手掛けているものの内，特に興味を抱いて研究を行っているデータベース・マシン開発へのHARPSの応用について述べる。

この研究は，HARPSの柔軟なエミュレーション機能を用いてデータベース・マシンのアーキテクチュアを研究するもので，この研究のために，ローカル・メモリーの増設等のHARPSの一部拡張も決定している。現在はアーキテクチュア設計のための基礎研究の段階であり，この諸結果に基づき，具体的なアーキテクチュアの提案をいくつか行い，HARPSによるエミュレーション等を通して問題点の把握を行いつつ，最適システムの設計へと収束させていく予定である。

この研究の基本思想は

1．ファイル・アクセスの並列化

2．処理の並列化

を押し進めることにより，関係形式データベースを効率良く実現するハードウェア，ファームウェアを開発することにある。この基本思想は

3．並列化の単位を関係中の各定義域(domain)に選ぶ。

ことにより，関係形式の特長を活かした並列化が可能となる。

同じような方向を目指したシステムとしてはRAP[2], RARES[3], STARAN[4]等があるが，並列アクセスの考えが弱く，処理の並列化も充分ではない。同一データの除去，データ管理，マルチ・ユーザーをも考慮したスケジューリングの問題等において，満足な解決がなされていない。この報告では，著者等が目指しているOPREC (Orthogonal Processor for Relational Computation) のアーキテクチュアの現時点での考え方を示し，OPRECの設計理論やマルチ・ユーザー状況下でのスケジ

ュール問題等を概説するとともに，データベース・マシンに対する要請と批判について考察する。

[2] DBMSにおける並列化の可能性

DBMSには大きく分けて3種類のモデルがあるが，これらは，ファイル・アクセス経路決定の3種類の方法と考えられる。これにDBMSの持つべき機能と探索手法を加えてまとめると以下のようなものが得られる。

A．ファイル・アクセス
  a．階層モデル
  b．ネットワーク・モデル
  C．関係形式モデル
B．機能
  a．ソート機能 } etc.
  b．マージ機能 }
C．探索手法
  a．ハッシュ・テーブル
  b．Balanced-tree，B-tree etc. } etc.
  c．連想記憶

データベース・マシンと呼ばれるものは，これらA，B，Cの内の1つないし2つ以上の項目に関し，特殊アーキテクチュアにより処理の高速化を図ろうというものである。しかし，この報告では，データベース・マシンはデータベース全体に関連するものを指すとし，従って，Aに関するもののみを考えることにする。

DBMSにおけるデータと数値計算におけるデータの性質を比較すると以下のことが言える。

1．数値計算では1語データが個別に取り扱われることが多いが，DBMSではデータが集合単位で取り扱われ，1つの集合中の個々のデータは一様な処理を受けることが多い。
2．数値計算ではデータ間に因果関係を持つことが多いが，DBMSでは処理のスケジュールに影響するような因果関係は個々のデータ間ではほとんどなく，集合間において統計的に言って弱い関係しかない。

以上の2点は上述のA，B，Cにおいて，並列処理の可能性が非常に高いことを意味している。1は並列処理の対象が充分多いことを意味し，2は並列化の際の競合や，並列化を阻む条件が少いことを保障する。1，2はまた，並列アクセスという数値計算の場合には使われることの少なかった手法が効果的であることを示している。

[3] データベース・マシン —要請と批判—

DBMSの3種類のモデルの内，階層モデルとネットワーク・モデルは，フォン・ノイマン型計算機の汎用OSのつくる記憶空間の構造に本質的に強く依存しており，純粋にDBMSの高効率化を図った結果の産物とは考えられない。これに対し，関係形式モデルは論理的に整った構造を持ち，推論的検索システムへの発展の可能性を秘め，表現能力が高く，高度な検索要求を容易に与えることができる。データベースの更新が容易で，サブモデルや応用システムが作り易く，データ独立性が優れている。

しかし反面において，（A）これを従来の汎用OSの作る記憶空間内で実現する際の効率の悪さと，（B）従来作られたデータベースが，階層モデルやネットワーク・モデルにおいて一部インバーティッド・ファイルを用意することで効率良く実現できたという事実ゆえに，本格的大規模データベースにおいて関係形式が採用されることは少なかった。（B）はしかしながら，今後関係形式の研究が進むにつれ状況が変化すると考えられる。問題は（A）であり，ここにデータベース・マシン研究の大きな動機がある。

データベース・マシンという考え方に対する批判が多い。それは次のような理由からである。

A．コスト・パーフォーマンスと技術の面から，特殊計算機を開発する意義があるか？
B．既存のデータベースを有効に利用可能か？
C．ホストとデータベース・マシン，あるいはデータベース・マシンと端末を結ぶチャンネル・スピードがボトルネックにならないか？
D．マルチ・ユーザーに対して対応し得るか？
E．新しいメモリー・ディバイスが開発されれば，データベース・マシンの構想が変化するのではないか？

これらの批判に対し，著者等は次のように考える。英文字は上述の批判のそれに対応する。

A．DBMSを始めいくつかの分野においては，計算機システムそのものが，計算をするものから，データベース的なものに変化しつつあ

り，将来主従関係が逆転することも考えられる。技術的には，マイクロコンピュータの発展が実現性を増している。

B．関係モデルそのものの問題である。むしろ今後この分野の研究が進むものと思われる。

C．検索コマンドの高度なマクロ化と中間結果の独自処理により解決できる。

D．2章で述べた理由により，数値計算に対するスケジューリングよりも容易である。効率向上が大きい。

E．OPRECは，例えばCAM（Contents Addressable Menory）の使用により，一層の高速性を得ることができる。

以上述べたように，データベース・マシン開発の意義は大きい。

[4] 関係形式データベース[5)]

関係形式データベースでは，情報ないしは情報の構造情報が表の形で表現される。

4.1．定義

集合列，$D_1, D_2, \cdots D_n$ に対し，直積集合 $\prod_{i=1}^{n} D_i$ の部分集合Rを関係と呼び，同じ $\prod_{i=1}^{n} D_i$ の部分集合の族𝕽を関係の型という。nを関係の型の次数といい $deg(\mathbb{R})$ で表す。Rの要素を組といい，Rの要素数を組数といい，$tuple(R)$ で表す。各 D_i は定義域と呼ばれ，$D_1, D_2, \cdots, D_n$ はすべてが異なる必要はなく，これらを区別するために $A_1, A_2, \cdots, A_n$ なる名前をつけ属性と呼ぶ。A_i は第i属性である。Rが $A_1, A_2, \cdots, A_n$ なる属性を持つことを $R(A_1, A_2, \cdots, A_n)$ で表現する。Rの属性全体を $\Omega(R)$ で表す。$A \subset \Omega(R)$ が $A = \{A_{i_1}, A_{i_2}, \cdots, A_{i_k}\}$ のとき，$\lambda(A) = (i_1, i_2, \cdots, i_k)$ とし，これをAのインデックス集合，i_n をインデックスと名付ける。関係Rが

$$R = \{r_i \mid i = 1, 2, \cdots, tuple(R)\}$$
$$r_i = (r_{i1}, r_{i2}, \cdots, r_{in})$$

なるとき，Rは第i行第j列に r_{ij} を書いた $tuple(R)$ 行n列の表によって書き表すことができる。このように表において，$r_i \neq r_j$ $(i, j$ $i \neq j)$ が言えるとき，同一データの重複がないといい，このときさらに，どの属性もモデルとしてそれ以上分割して考えられることがないとき，この表現を第1正規形という。関係形式データベースDとは関係の型の集合であり，ある時点でのデータベースDは関係の集合である。Dに現れるすべての属性の集合を $\Omega(D)$ で表す。

4.2．集合演算と関係計算

関係は集合であるから，∪（集合和），∩（集合積），−（集合差），×（直積）等の2項集合演算が可能である。

関係計算にはCoddの提唱になる関係代数があるが，この報告ではそれとは異なるより一般的な表現法を示す。

2つの関係の型 $R(A_1, A_2, \cdots, A_n)$，$S(B_1, B_2, \cdots, B_m)$ に対し，

$$B_i = f_i(A_{i_1}, A_{i_2}, \cdots, A_{i_{g_i}})$$

により，RをSに移す関数Fを考えることができる。Fを

$$F \equiv (f_1(1_1, \cdots, 1_{g_1}), \cdots, f_m(m_1, \cdots, m_{g_m}))$$

で表現する。$F = (1, 2)$ は第1，第2属性への射影を意味する。

関係Rに対し，RからRの部分集合への写像Cを考えることができ，制約条件と呼ぶ。Cは一般に，

$$C \equiv (\bigvee_j (\bigwedge_i g_i^j(i_{i_1}^j, \cdots i_{i_{g_i}}^j) \theta\ "v_i^j"))$$
$$(\bigwedge_i h_i^{i'}(i_{i_1}^{i'}, \cdots i_{i_{g_i}}^{i'}) \theta\ k_i^{i'}(i_{i_1}^{i'}, \cdots i_{i_{g_i}}^{i'})))$$

の形をしている。∧，∨ はand，orを表し，"v"は具体値を表す。C(R)に対し，$g(i) \theta$ "v" をRのセレクションと呼び，$h(i) \theta k(i')$ をRのリストリクションと呼ぶ。ここで θ は関係演算子で $=, >, <$ 等である。例えば $\{1, 2, 3\} \subset \lambda(\Omega(R))$ に対して，

$$C(R) = ((1 = "\alpha") \wedge (2 = 3))\ R$$

は，Rにおいて，第1属性が値"α"で，第2属性と第3属性の値が等しいような組を求めることを意味する。Fは関係に対して適用でき，関係Rに対して，F，Cを同時に適用することを $(F; C)R$ で表す。$(F; C)R$ を関係計算と呼び，$(F; C)$ をFC作用素，Rを台集合と呼ぶ。

$$(F';C')R \times (F;C)S$$

を

$$((F';C')\times(F;C))\ R\times S$$

で表す。$(F';C')$, $(F;C)$ は，それぞれ台集合が R，S の作用素である。これを台集合が R×S の作用素に拡張する操作を σ で表わすと，

$$\begin{aligned}&(F';C')\times(F;C)\\&=\sigma(F';C')\cdot\sigma(F;C)\\&=(\sigma[F']\cdot\sigma[F]\ ;\ C'\wedge\sigma[C])\end{aligned}$$

ただし，$((1,2,3);T)\cdot((1/2);T)=((1/2);T)$ 等とする。T は無制約であることを表す。例えば $deg(R)=2$，$deg(S)=2$ とし，$(F';C')=(1/2;\ 2="5")$，$(F;C)=(2;\ 1=2)$ とすれば，$\sigma(F';C')=(1/2,3,4\ ;\ 2="5")$，$\sigma(F;C)=(1,2,4\ ;\ 3=4)$ となり，

$$\begin{aligned}&\sigma(F';C')\cdot\sigma(F;C)\\&=(1/2,3,4\ ;\ 2="5")\cdot(1,2,4\ ;\ 3=4)\\&=(1/2,4\ ;\ 2="5"\wedge 3=4)\end{aligned}$$

となる。ついでながら，これは，

$$=(1/"5",\ 4\ ;\ 2="5"\wedge 3=4)$$

と簡略化できる。

関係代数のジョイン演算

$$R[A\theta B]S=\{(r,s)\mid P_A(r)\,\theta\,P_B(s)\quad r\in R,\ s\in S\}$$

$A\subset\Omega(R)$，$B\subset\Omega(S)$

P_A：属性集合 A への射影：$(\lambda(A);T)$

は，

$$(\mathrm{I}\!\mathrm{I}\ ;\ \lambda(A)\,\theta\,\sigma[\lambda(B)])\ R\times S$$

で表現される。$\mathrm{I}\!\mathrm{I}$ は恒等写像を表す。

関係代数における商

$$R\div S=\{g\mid (S.g)\in R,\ {}^{\forall}S\in S\}$$

は，$S=P_{A_1}(S.g)$，$g=P_{A_2}(S.g)$ とし，

$$(\lambda(A_2)\div\delta[\lambda(\Omega(S))];\lambda(A_1)=\delta[\lambda(\Omega(S))])\ R/S$$

と表現する。δ はインデックスにプライム "′" を 1 つ付加する操作を表す。これにより，各インデックスが，台集合の一般形

$$\Pi R/\Pi S/\cdots$$

において，どの部分のインデックスであるか知ることができる。

関係計算 $(F;C)$ による記法は関係代数に比し複雑に見えるが，これは，関係代数が概念的であるのに比し，関係計算 $(F;C)$ が計算機処理を念頭に置いているからである。

関係計算 $(F;C)$ に対し，以下の関係が成り立つ。

a. $(F';C')\times(F;C)=\sigma(F';C')\cdot\sigma(F;C)$

b. $(F';C')(F;C)=(F'F\ ;\ C\wedge C'[F])$

c. $F(F_1\div F_2)=(FF_1,\ F_1\div F_2)$

$(F_1\div F_2)F=F_1F\div F_2F$

$(F_1\div F_2)(F_3\div F_4)=F_1(F_3\div F_4)\div F_2(F_3\div F_4)$

$=(F_1F_3\div F_2F_3,(F_3\div F_4))$

ここで，$C[F]$ は C なる制約を，F を作用する前のインデックスに関する制約に直すことを意味し，例えば $(3=4)[5,2,1,6]$ は $(1=6)$ である。

このことから，任意の関係計算は，

$$(F_1\div F_2,(F_3\div F_4,(\cdots));C)\ R_1/R_2/\cdots$$

$$R_k=\prod_{k}R_{ki}$$

の形の標準形に変形可能であることがわかる。この計算は，プライム "′" の多くついているインデックスから順に実行して行くという制約があるだけで，プライムの個数の等しいインデックスに関する実行の順序は自由である。関係計算を標準形に直すことは，$(F;C)$ 作用素の簡略化やスケジューリングにとって重要である。

関係 R がデータベースに表形式で記憶されているとき，R を一次関係と呼び，R が関係計算の結果生まれるものであるとき，R を二次関係と呼ぶことにする。

4.3. 関数従属性

$A,B\subset\Omega(R)$ のとき，R の任意の関係 R，R の任意の 2 組 r_1,r_2 に対して，

$$P_A(r_1)=P_A(r_2)\Rightarrow P_B(r_1)=P_B(r_2)$$

が成立するとき，B は A に関数従属であるといい，$A\to B$ と表す。

$A \to \Omega(R)$で，任意の$A' \subset A$に対して$A' \twoheadrightarrow \Omega(R)$が成立するとき，Aはキャンディデート・キーと呼ばれる。Rのキャンディデート・キーの集合をK(R)で表わす。K(R)の中に属性A_iを含むキーがないとき，A_iはプライムでないと言われる。任意に選んだ$A \in K(R)$をプライマリー・キーと呼ぶ。ある$A \in K(R)$に対し，任意のプライムでない属性$B_i \in \Omega(R)$と任意の$A' \subset A$に関して，$A' \twoheadrightarrow B_i$ならばRはAに全関数従属であるといい，このとき，Rはプライマリー・キーに対しても全関数従属である。

全関数従属になるように関係を分解し得られたモデルを第2正規形と呼ぶ。$A \to B$で$B \to C$のとき，CはAに推移従属であるという。第2正規形で，任意のプライムでない属性$B_i \in \Omega(R)$が，ある$A \in K(R)$に推移従属でない場合，Rは第3正規形であるという。第3正規形では，プライマリー・キーに属する属性が，プライマリー・キーに推移従属することがあり，これをキー破壊と呼ぶ。分解された関係は，ジョイン演算により復元できる。この点において，ジョインは重要である。

[5] アーキテクチュア

既にRAP，RAREs，STARAN等が提案されているが，これらに関し以下の問題点がある。

a．演算の結果生じた2次関係は1次関係と同様に取り扱い可能であるべきである。
b．マルチ・ユーザー状況での考察が乏しい。
c．並列アクセスの利点を充分利用していない。
d．効率に関する評価に乏しい。
e．ジョイン演算の高速化に関する工夫がない。
f．同一データの除去，ソート，関係の除算の実行が遅い。
g．可変語長を目指しているが充分でない。

等の問題がある。

OPRECはこれらの点を考慮し，定義域毎の並列アクセスと並列処理を基本思想として考えられている。

OPRECは検索要求解析システムのもとで働くがこれを図1に示す。検索要求解析システムの内，スキーマプロセッサの一部を除いては本報告では，このシステムには触れない。スキーマプロセッサは関係計算の解析を行い，これを標準形に変形し，演算順序の解析を行った後，他のユーザーの関係計算に対する結果と比較し，同時処理の可能性を解析した後，この結果に基づいて各関係計算を基本演算に分解し，OPRECに送る。OPRECを図2に示す。関係は，1つ1つの組がD→MDSW→MEM→PMSW→MP→PMSW→MEM→MDSW→Dの多重経路を各定義域毎に分けられて通過していく間に処理される。

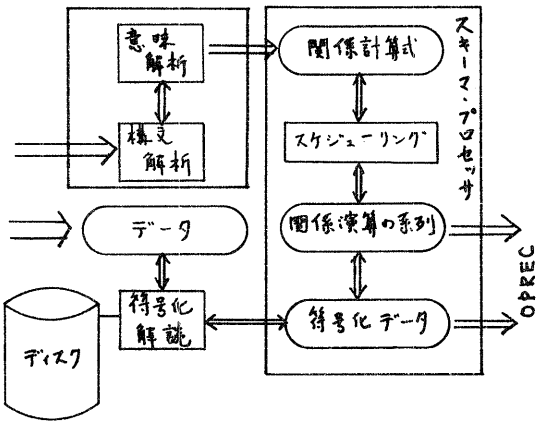

図1：検索要求解析システム

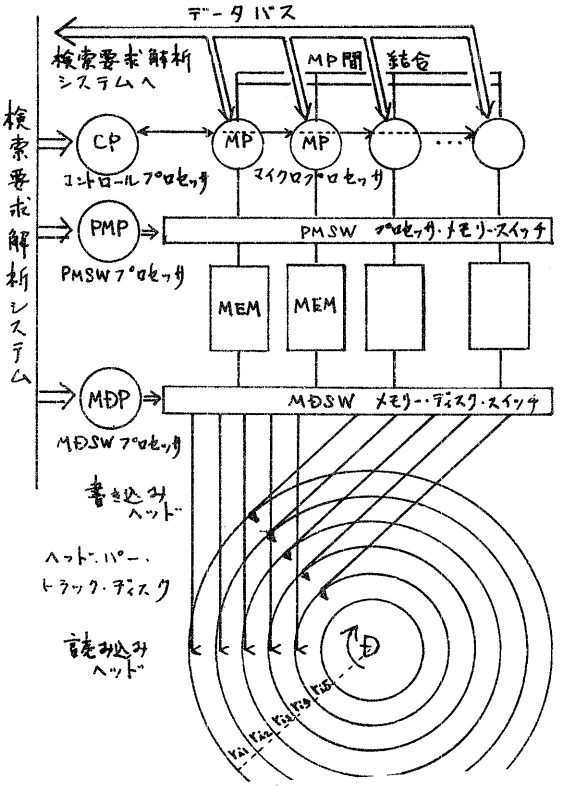

図2：OPRECの構成図

この処理は回転記憶装置Dの回転に遅れることなく繰り返される。処理の単位は1組つづではなく、バッファ・メモリーMEMのサイズをBとしてB語づつまとめて処理される。関係Rは基本的には回転記憶装置の1トラックに1定義域が割り当てられ、図3のように記憶される。図3におけるMEMのチェインはMDSWとPMSWで作られ、n_B台のMPに割り振られる。読み込みヘッドと書き込みヘッドを図のように配置することにより、データの更新が可能である。読み込みトラックと書き込みトラックが異なることもある。n_Bは通常1である。

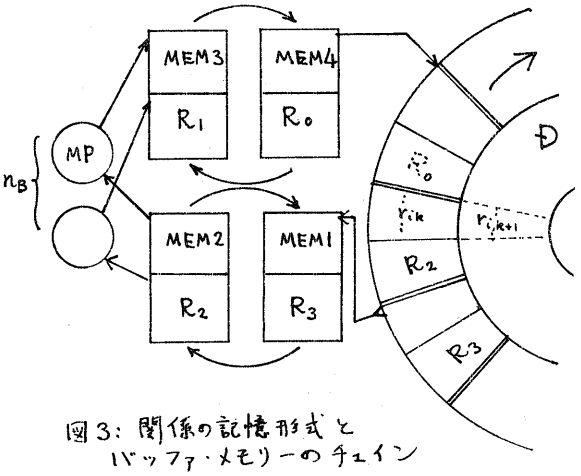

図3: 関係の記憶形式と
バッファ・メモリーのチェイン

ディスクは複数台に分かれてもよい。$R=R_1 \cup R_2$, $R_1 \cap R_2 = \phi$のとき, R_1とR_2は2台のディスクに分配可能であるが, $A \cup B = \Omega(R)$, $A \cap B = \phi$なるA, Bを分配することは許されない。これはディスクが異なれば回転の同期がとれないからである。定義域Dのデータ長が, OPRECの基本語長のW_D倍であるとき, Dを記憶するためのトラック数をTr_D, 処理用バッファ・メモリー数を$4M_D$, 処理用プロセッサ数をP_Dとすると, $P_D = M_D = Tr_D = W_D$とすることにより可変語長のデータが取り扱い得る。$P_D = M_D = n_B Tr_D = nW_D$ $(n = k \cdot n_B)$による処理形態を(n, n_B)way方式, (n, n)方式をnway方式と呼ぶ。これらは処理スピードを向上する。OPRECは関係Rの処理結果に従って, Rの各組にマークをつける機能(例えばR(A1, A2)に対し("mark", 1, 2; 1>2))を持つが, マーク用には関係RのためのP_R, M_R, Tr_Rとは別に, P_M, M_M, Tr_Mが取られる。(n, n_R)way方式では$P_M = M_M = n_B Tr_M = nW_M$で, W_Mは自由に選べる。

R, Sを1次関係ないしはディスクに既に表形式で生成されている2次関係とする。OPRECに対するコマンドは, (F;C)R, (F;C)R×S (F÷F';C)R/S, DUP(R)等である。DUP(R)はRにおける同一データの除去を行う。これらを一般に(F;C)Rと表わすと, d(F;C)でF, Cのいずれかに関係するインデックスの集合を表すとき, MDSWは, Pd(F;C)Rを行い, 選ばれた定義域に対するトラックを空いている(MP; MEM1, MEM2, MEM3, MEM4)の組に結びつけ, MEM1とこのトラックを結合する。(MP; MEMi)の組はPMSWによって作られる。処理結果が, もとのトラックと異なるトラックに書き込まれることも許し, MEM4と, 書き込みトラックの結合がMDSWで行われる。(F, C)Rの実行中のある時点では, MEM1に対し読み込みトラックからのデータ転送が起っており, 同時にMEM4から書き込みトラックへのデータ転送がおこっている。一方(MP; MEM2, MEM3)を1つのプロセッサ・システムとする密結合を持ったアレイプロセッサ・システムでは, MEM2に書かれたB語のデータに対する(F, C)Rの実行がプロセッサ間の密結合を使って行われている。処理結果はMEM3に書き込まれていく。これらの過程は同時に終了し, MEM1とMEM2, MEM3とMEM4が交換されて次のB語の処理に進む。検索要求に対する最終結果は, MEM3に書き込まれないで, 検索要求解析システムに並列に送られる。

(F, C)Rは1回転。Dlをトラック長として(F, C)R×Sは Dl/B回転で処理される。(F÷F'; C)では, 最初のDl/B回転でPd(F÷F;C)(R), Pd(F÷F;C)(S)の同一データの除去が行われ, 次のDl/B回転でRに対する中間結果R'の内, 中間結果S'に対し条件を満足するものに対し, R'に付加したマーク・エリアに1を加えていく。次のDl/B回転で直前のRの中間結果R"に対し, Pd(F)に対し同一データの除去を行うと共に, それらのマーク・エリアは加え合わす。最後の一回転でマーク・エリアがtuple(S')となっているものを探す。結局3(Dl/B)+1回転必要である。

OPRECはその他, カウント命令, ソート命令Min, Max命令等を持つものとする。Min, Max

は、対応する定義域を k として

$$Min(k;R) = (\lambda(\Omega)\div k',\ k \le k')$$
$$Max(k;R) = (\lambda(\Omega)\div k',\ k \ge k')$$

に等価である。

次に、B語のプロセッサ・アレイによる処理について述べる。簡単のために、$(F,C)R$ に対して、リストリクション

$$(1;2>3)R$$

を考える。バッファは、Rの第1、第2、第3属性を図4のように持っているものとする。

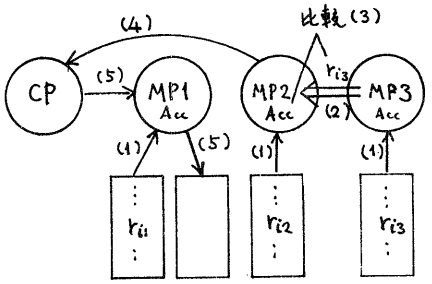

図.4 : $(1;2>3)R$ の実行

CPから、MP1, MP2, MP3に対し、データのロードが指定される。3台とも同じ番地を指定する。CPはMP3に対しMP2へのデータ転送指令を送り、次にMP2に対しAccと転送データの比較を指示する。比較結果はCPに送られ、これによりCPはMP1にデータ書き込みを指示する。以上が1サイクルで、読み込み番地が1加えられた後、次のサイクルに移る。

$(F;C)R\times S$ に対して、簡単のために、

$$(1,4;2=3)R(A_1,A_2)\times S(B_1,B_2)$$

の実行を考える。これを図5に示す。転送用ポインタ P_x は最初データの先頭にある。比較用ポインタ P_c をMP1, MP2では P_x に、MP3, MP4では P_x+1 にセットする。MP2, MP3で P_x のデータを交換する。このデータをXとする。MP2, MP3においてXと P_c を比較し、比較条件を満足するとき、各MPのステータスを1にする。MPアレイのステータスはCPによって常に監視されており、MP2, MP3のいずれかが1となったとき、それぞれに対応して P_R と P_S の2種

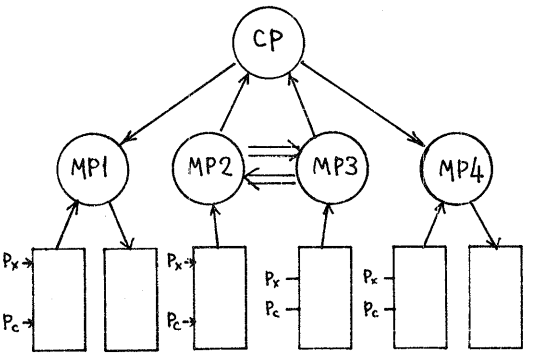

図5. $(F;C)R\times S$ の処理.

$((1,4);2=3)R(A_1,A_2)\times S(B_1,B_2)$

のパルスをMP1, MP4に送る。P_R, P_S の両方が送られることがある。MP1, MP4では、MP2, MP3に対応して、P_x, P_c が変化しており、P_R が送られるとMP1は P_c のデータを、MP4は P_x のデータをMEM3に書き込み、P_S の場合は、P_x, P_c が逆になる。MEM3が1杯になったときは、PMSWとMDSWによって新しいMEM3, MEM4と新しいトラックがこのMPに割りあてられる。これにより、トラック数はどんどん増すが以後の回転の際に中間結果が使用するトラック数の最少化が行われる。次に P_c が P_c+1 となり、P_c がバッファ・サイズをこえたとき、$P_x=P_x+1$, $P_c=P_x$ となる。

(n, n_B) way方式では、1度に他のn個のMPから送られたデータと P_c のさすデータの比較を行う。一回の比較はn倍の時間がかかるが、同時に n^2 倍の比較ができ、時間は $1/n$ となる。

[6] OPRECの設計理論

OPRECの設計に必要な式を求める。先述のものを含め、以下のような記号を導入する。

関係Rに対して、

$deg(R)$: 次数 , $tuple(R)$: 組数

$width(\theta) = W_\theta$, $width(R) = \sum_{\theta\in\Omega(R)} W_\theta$

回転記憶装置に対して、

θ_l : トラック長 (word)

θ_w : トラック数

τ：1回転に要する時間（sec）

バッファに対して，

B：バッファ・サイズ（word）

N_{MEM}：バッファ・メモリーの個数

プロセッシング・エレメントに対して，

C：データの比較，演算に要する時間のオーダー（sec）

Δ：プロセッサ間の1wordのデータ転送時間（sec）

C'：2つの定義域にあるデータ間の比較に要する時間のオーダー（sec）$C' \fallingdotseq C+\Delta$

N_p：プロセッサの数．

次のようなOPRECのモデルを考える。

cφ：1つのトラックには1つの定義域のみを記憶する。

C1：$Dl > \max_R tupl_e(R) \gg B$

C2：$Dw, N_{MEM}, N_p \gg \max_R deg(R)$

R，Sを一次関係ないし中間結果として表形式で記憶されている二次関係とし，T_1を(F,C)Rの実行に要する時間（sec），T_2を(F,C)R×Sの実行に要する時間とすると，

$$T_1 = \tau \qquad (1)$$

$$(B/Dl)\tau > BC \rightarrow \tau/Dl > C \qquad (2)$$

$$T_2 = \tau\,(Dl/B) \qquad (3)$$

$$\tau\cdot(B/Dl) \fallingdotseq (C/2)B^2 \rightarrow (\tau/Dl) \fallingdotseq (C\cdot B/2) \qquad (4)$$

が得られ，

$$T_2 = Dl^2\cdot C/2$$

となり，T_2はτ，Bに依存しない。C，Dlはだいたいの値が決まってしまうので，τ，Bに関する条件を求めると，

$$\tau > C\cdot Dl$$

$$(\tau/B) \fallingdotseq C\cdot Dl/2$$

$$B > 2$$

となる。$C = 10^{-7}$，$Dl = 10^4$とすると，

$$\tau > 10^{-3}_{sec},\quad T_2 = 5\ sec$$

となる。ディスク使用の際は，$\tau = 10^{-2}$とすると，

$$B \fallingdotseq 20$$

が得られる。

cφを変えて(n:1)wayとすると，

$$T_{2,(n,1)} = T_{2,1}$$

$$(\tau_{(n,1)}/B_{(n,1)}) = n(\tau_1/B_1)$$

$$N_{p(n,1)} = n\cdot N_{p1}$$

$$N_{MEM(n,1)} = n\cdot N_{MEM1}$$

となり，パーフォーマンスは良くない。

nwayでは，

$$T_{2,n} = n^{-1}\cdot T_{2,1}$$

$$\tau_n/B_n = \tau_1/B_1$$

$$N_{pn} = n\cdot N_{p1}$$

$$N_{MEM\,n} = n\cdot N_{MEM1}$$

となる。

以上のことから，Bは，τが実現可能程度に小さくすればよいとことがわかる。τの小さいCCDや磁気バブルを使用する試みがあるが，これらのデータ長が$tupl_e(R)$に比し小さい場合は，Bを高速メモリより安価なメモリに置き換えるという効果に終る。このことは，しかしながら重要であり，現在検討中である。

以上のモデルでは，処理時間は$tupl_e(R)$に関係しないが，回転記憶装置のバッファ・サイズ毎に初期位置があれば，

$$T_1 \propto tupl_e(R)$$

$$T_2 \propto tupl_e(R)$$

となる。

さらに，

cφ'：回転記憶装置上にRを格納する際，1トラックのDl/B個のブロックに均等に「$tupl_e(R)$/B」個のブロックを配分する。残りのブロックは別の関係に使用する。(F,C)Rの処理では回転記憶装置がk_1ブロック進む間に1ブロックの処理が終了するとし，(F,C)R×Sではk_2ブロック進む時間を要するとする。($k_2 >> k_1$)処理はnwayで行う。

とすると，

$$T_1 \fallingdotseq \lceil tupl_e(R)\cdot k_1/(n\cdot Dl)\rceil\cdot\tau$$

$$k_1\cdot\tau/Dl > C$$

$$T_2 \fallingdotseq \frac{C \cdot k_2}{2n} \cdot tuple^2(R)$$

$$\tau/B = n \cdot (C \cdot Dl)/2$$

が得られ、

$$T_2 \propto tuple^2(R)$$

とできる。

通常のn wayにおいてCAMをバッファとして使用するならば、

$$T_2 = Dl^2 \cdot C / (n \cdot B)$$

となり、非常に効率が上がる。ただし、

$$\tau = C \cdot Dl$$

でなければならない。

[7] スケジューリング

関係計算の標準形

$$(F_1 \div F_2\ (F_3 \div F_4\ (\cdots))\ ;\ C)\ R_1 / R_2 / \cdots$$

$$R_k = \prod R_k^{k_i}$$

が得られたならば、Fの各レベルに関して簡略化を行い、Cについては"/"に関する各レベルで簡略化を行う。その結果C=F（Fは偽を表す）となれば、この検索の実行は必要ない。レベル解析から、いくつかの処理順序に関する制限が得られる。これを商条件と呼ぶ。単一ユーザーの場合は、$R_1/R_2/\cdots$の中に同じ関係が現れていないか探し、商条件を満たす範囲でそれらに対する処理を同時に行うようにする。マルチ・ユーザーの場合も、他の関係計算の中に同時実行可能なものがないかを探す。このアルゴリズムは、商を含まない場合は簡単で台集合を関係すべてに対して決められた順番に並べ換えた直積形に変形し、他ユーザーの関係計算中に、同じような並びがないかを調べればよい。除算を含む場合は、並べ換えの際に商条件が制約として入ってくるだけで、同じようにスケジュール可能である。集合の順番を決めたならば、(F,C)作用書を各2項ないし1項演算に分解する。詳細は別の機会に述べる。

[8] データベース・マシンの課題

CAMを使用しないデータベース・マシンは次のような課題を残している。

A. 同一データの除去
B. ソートの高速化
C. 符号化と符号の解読法

ここではOPRECに関し、A,Bを考える。

Bに関しては回転記憶装置の外部記憶に対する外部ソート問題の理論に依ればよい。その結果は満足し得るものではなく、特殊な工夫が今後必要になる。この問題はCAMを使用した際に1問題となる。

同一データの除去は、ほぼ6章のT_2だけの時間を要し、大きな損失である。除去はその都度行う必要はないが、中間結果をそのままにしておくと、冗長度はジョインの度に指数関数的に増加する。この問題に対し、いつ冗長なデータが発生し得るかが分れば都合がよい。

関係Rに対し、dup(R)をRに同一データの重複があればϕ、なければ1をとる述語とする。cand(A)をAがRのキャンディデート・キーであるという述語とする。これはcand(A;R)とも書く。univ(A)を$P_A(R)$の要素が1個であるという述語とする。これらを使って、非冗長性の伝搬規則を以下に示す。

a. $dup((\mathbb{1}, C)R) = dup(R)$

b. $dup(R[A=B]S) = (dup(R) \wedge dup(S))$

c. $dup(P_A(R)) = (cand(A) \wedge dup(R))$

d. $cand(A) \wedge cand(B) = 1$ のとき、
$tuple(A) = tuple(B)$

e. $dup((F;T)R) \geq cand(dom(F)) \wedge inv(F) \wedge dup(R)$

f. $dup((F;C)R) \geq dup((F;T)R) \wedge dup((\mathbb{1};C)R)$

g. $cand(A;R \times S) \geq cand(A;R) \wedge dup(R) \wedge \{R \to S\}$

$cand(A;R \times S) \geq cand(A;R) \wedge dup(R \times S) \wedge \{R \to S\}$

ただし、inv(F)はFに逆関数が存在するという述語である。伝搬規則はこの他にも多くあるものと考えている。これらを見つけることも、本研究の一つのテーマである。

これらの論理不等式を(F;C)に適用することにより、どこまで非冗長性が保存されているかがわかり、保障が切れた時点で除去を行えばよい。むやみに除去を行うよりは効率が良くなる。cand(A)を決定するアルゴリズム、つまり、4章におけるK(R)を求めるアルゴリズムは存在する。

[9] 結論

この研究は、現在、基礎理論の研究が中心になっているが、近い将来、HARPSによるエミュレーション実験に移行する予定である。

関係形式データベースは、ファイル・アクセスのみならず、OPRECのスキーマ・プロセッサに相当する部分に関しても種々の工夫が考えられるモデルであり、自然言語との関連もつきやすく、今後、さらに重要性を増すものと考える。

OPRECの基本思想である並列アクセス、並列処理は、CAMの使用やプロセッサの仕様の工夫、回転記憶装置の階層化等、今後さらに深い検討を要する。

[謝辞]

本研究は、その一部を、昭和51年度、文部省特定研究(2)「情報システムの形成過程と学術情報の組織化」における研究活動の一部として、行った。貴重な御意見を頂いた諸先生方に感謝の意を表します。

Bibliography

1) Tanaka,Y., and Miyashita,K., et al. HARPS (Hokkaido University Array Processor System) : A new hierarchical array processor system. Proc. 2nd EUROMICRO Symp. 1976, North-Holland.

2) Ozakarahan,E.A., Schuster,S.A., and Smith,K.C. RAP-An associative processor for data base management. Proc. AFIPS 1975 NCC, Vol.44, AFIPS Press, Montvale, N.J., pp.379-387

3) Lin,C.S., Smith,C.P., and Smith,J.M. The design of a rotating associative memory for relational database applications. ACM Trans. on Database Systems, Vol.1,No.1, 1976, pp.53-65.

4) Davis,E.W. STARAN parallel processor system software. Proc. AFIPS 1974 NCC, Vol.43, AFIPS Press, Montvale, N.J., pp.16-22.

5) Codd,E.F. A relational model of data for large shared data banks. Comm.ACM 13,6(June 1970), 377-387