# データ分配ネットワークを用いたデータベース計算機の構成法

# Data Base Machine Architecture using Data Partitioning Network.

小田　泰充
Yasumitsu Oda
日本電信電話公社　武蔵野電気通信研究所
Musashino Electrical Communication Laboratory, N.T.T.

## 1. まえがき

近年の情報化社会の発展にともない，データベース（以後単にDBと記す）に対する高機能化，高速化の要求が高まっている。しかし，従来のノイマン型計算機では，① 高機能化を果そうとすると，ソフトウエアの負担が増大する。② 高速化を果そうとすると，補助記憶と主記憶および主記憶とCPUの間のデータ（および命令）の転送が隘路となる。という欠点があった。

これに対し，VLSI技術の進歩により急速に価格が低下しつつあるハードウェアを利用して，DB専用のアーキテクチャーをもった計算機（Data Base Machine 以後単にDBMと記す）を作成しようとする試みが活発に行われている。その主なものに

(1) セル論理装置型のもの[1],[2]
RAP, CASSM, RARES, DBC
DIRECT, EDC[5]

(2) ハッシュビットマップを用いたもの
CAFS[3], LEECH

(3) 動的ビットマップを用いたもの
SPIRIT[4]

(4) 分散データベース指向のもの
DIALOG[6]

(5) 関係演算専用プロセッサを用いたもの
DSDBM（ソートエンジン，サーチエンジン）[9]
Systolic Array による関係代数演算器[7]
データ圧縮ソートによる関係代数演算器[8]

などがある。

これらのDBMは，いずれもCoddが提唱した関係モデル（RDB）を効率よく実現する目的で（少なくとも目的の1つとして）開発されている。しかし，(1)のタイプのものは，検索，更新，挿入，削除（以後これを基本演算と呼ぶ）は効率よく実行するが，RDBの特長である，JOIN, PROJECTION などの集合と集合の交差的演算（これを基本演算と区別して，集合演算と呼ぶことにする。）には向いていない。(2)のタイプのものでは，ハッシュ技法を集合演算の前処理としてのみ利用しており，演算そのものは，ホスト計算機で実行しなければならない。(3)のものでは，ビットマップを作成する手間が余分にかかることと，場合により作成したビットマップが巨大になるという欠点がある。

(4)，(5)のものは，いずれも集合演算を高速に実行するための専用ハードウェアをそなえている。特に，(5)のものは，$O(n)$（$n$：タップル数）で集合演算ができる。しかし，これらのハードウェアをm台用意し，m並列入力を行っても，これらを単に，直列，並列，あるいは網状に結合するだけでは，$O(n)/m$ とすることはできない。

1つの解決策は，m並列入力を直列入力に変換して集合演算器に入力することであるが，そのためには，読み出し速度のm倍の転送速度と，それに応じて高速動作する集合演算器が要求されるので，この方法には限界がある。

そこで，ここでは，基本演算ではセル論理装置を用い，集合演算では，データの入力速度に追随して集合演算を行う集合演算器を用い，こ

れとセル論理装置を，データ分配ネットワーク（Data Partitioning Network 以後DPNETと呼ぶ）で結合することにより，セル論理装置の並列読み出しに応じた並列度で，基本演算も集合演算も行えるDBMを提案する。本稿のDPNETは，セル論理装置から集合演算器へのデータ転送時間を利用して，一種のダイナミッククラスタリングを行うものである。

2章でシステム概要を，3章でシステム構成要素を述べた後，4章で評価と考察を行う。

## 2. システム概要

### 2.1 検討方針

本稿で提案するDBMは，次の方針のもとに検討した。

(1) RDB用DBMとする。

(2) DBの規模としては，ギガバイトオーダとするが，個々のベースリレーションは数十メガバイト以下のものが大部分であるとする。

(3) DBの格納媒体は，価格の点から磁気ディスクを前提とする。

(4) 同一ベースリレーションは，同一シリンダ群に集める。

(5) 言語インターフェイスは，関係代数レベルとする。（表2-1）

(6) シリンダ群の並列読み出し速度に応じた処理速度で，基本演算も集合演算も行えるDBMとする。

(7) DBの完全性チェック，Queryの最適化はソフトで行う。

(8) 同時アクセス制御は，関係表レベルのロックで対処する。（本稿では，これについては ふれない。）

### 2.2 基本的考え方

ここでは，DPNETを採用した基本的考え方を述べる。

今仮に，表2-1のOP2タイプの演算をP台のプロセッサを利用し，並列処理することを考える。この演算を，OP2($R_S$, $R_T$)と記す。ここで，$R_S$をソースリレーション，$R_T$をターゲットリレーションと呼ぶ。$R_S$, $R_T$が各々mP個，nP個のセル（回転メモリの1周期分のメモリ，この場合1セル＝1トラック）に格納されており，1プロセッサが1セルの$R_S$と1セルの$R_T$のOP2を実行するのに$T_c$時間かかるものとする。

このとき，$R_S$, $R_T$が図2-1のごとく，セル単位にP個にクラスタリングされている場合とそうでない場合を考える。

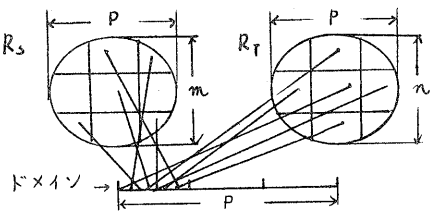
(a) クラスタリングなしの場合

$R_S$, $R_T$の全セル同志の比較が必要になりその処理時間$T_1$は，次式となる。

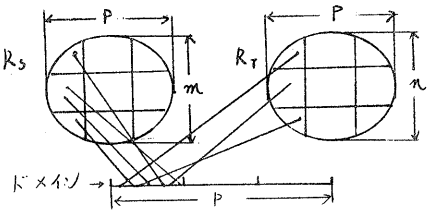
$$T_1 = \frac{mP \cdot nP}{P} T_c = m \cdot n \cdot P \cdot T_c$$

表2-1 提供する関係代数演算

| 演算タイプ | | 関係代数演算 |
|---|---|---|
| 基本演算 | OPφ | RESTRICTION, INSERTION<br>UPDATE, DELETION |
| 集合演算 | OP1 | PROJECTION<br>AGGREGATION OPERATION<br>(max, min, average<br>sum, cardinality count) |
| | OP2 | JOIN (implicit JOIN, explicit JOIN)<br>UNION, DIFFERENCE<br>INTERSECTION |
| | OP3 | DIVISION |




(a) クラスタリングされていない




(b) クラスタリングされている

図2-1 クラスタリング

(b) クラスタリング有りの場合

他のクラスタに属するセル同志の比較は必要でなくなるため処理時間 $T_2$ は次式となる。

$$T_2 = \frac{\sum_{\text{クラスタ数}} m \cdot n}{p} T_c = m \cdot n T_c$$

このように，クラスタリングされていれば，処理時間を $1/p$ に短縮することができる。常に図2-1(a)のごとくBをクラスタリングして格納できればよいが，リレーションは2つ以上の属性に対して定義されており，どの属性に対しても関係演算が定義できるので，すべての属性に対しそれを行っておくことは出来ない。

そこで，処理時間を $1/p$ に短縮するためには演算のたびに，ダイナミックに，遅延なく，図2-1(a)の状態を作成することが要求される。

本稿で提案するHPNETは，これを補助記憶から集合演算器へのデータ転送時間を利用して行うものである。

クラスタリングの手法としては，ソート技法を使用する方法やヒストグラムを知って分割した結果が平均化するように範囲を決めてやる方法[10]があるが，前者はソーティングに $O(n)$ が要求されること，後者はヒストグラムに関する知識が必要となり，しかも演算は部分集合に対して行なわれることが多いため，全体のヒストグラムに対する知識では役立たなくなる恐れのあること，などの問題点がある。そこでここでは，ハッシュ関数を通じてダイナミッククラスタリングを行う方法を採用した。ハッシュ技法を適用する際新たに発生する問題点として，

① データの分布を本当にランダムにすることができるか？

② クラスタ間の大小比較が不可能になるが関係演算への影響はないか？

がある。①に対しては，入力データの性質によるため，数種のハッシュ関数を用意し，ハッシュ関数を選択できる機能をもたせることにより対処する。②については，関係代数が集合に対して定義されているため，大小比較が必要となるのは θ-JOIN のみであり，他の演算では問題ない。この θ-JOIN が実用上必要となるのは極めてまれであるので，HBMとしてのハード的サポートは行なわないことにする。

なお，ハッシュ関数を用い，ランダムなデータ分布にしてから，データの分割を行い集合演算を行うという考えは，D.E. Shaw [11] にも見られるが，彼は関係表の読み込み時間は表の大きさと無関係であるとし，1つの集合演算器だけを用いた場合について論じているので，本稿の

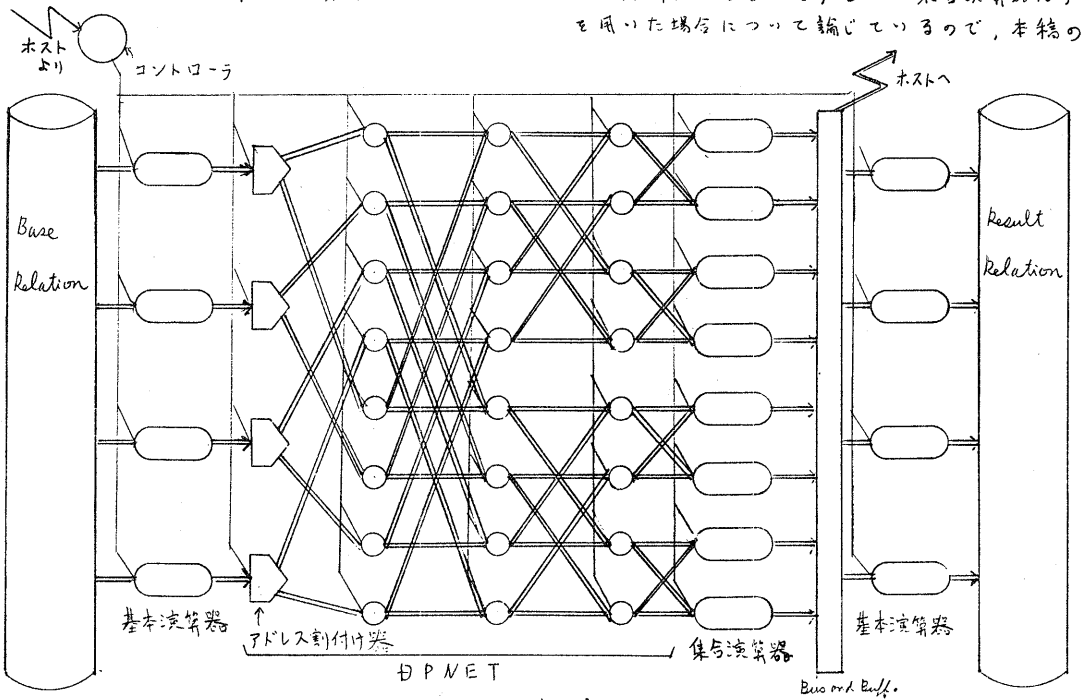



図2-2 システム構成図

場合（表の大きさと読み込み時間は比例，演算器は複数台並列処理）とは，大きく異なる。

## 2．3　全体構成

図2－2に，DPNETを用いたDBMのシステム構成図を示す。以下，全体の動作概要をOP2（$R_S$, $R_T$）演算を例に説明する。

① コントローラは，基本演算器に$R_S$の送出を，DPNETに$R_S$の転送準備を，集合演算器にOP2演算の準備を，それぞれ指示する。このとき，$R_S$送出と同時にRESTRICTION演算が可能なら，その旨基本演算器に指示を与える。

② 基本演算器は，補助記憶から関係表を読み込み，集合演算器が必要とするタップル（または，タップルの必要な部分のみ）を選択し，DPNETに送出する。

③ DPNETでは，演算対象となる属性をキィとしてハッシュ関数を計算し，その結果から，タップル送出先の集合演算器を決定して，当該集合演算器にタップルを送り込む。

④ 各集合演算器は，DPNETから送られて来るタップルを，いずれかの集合演算器がオーバーフローするまで，蓄える。

⑤ コントローラは，基本演算器から$R_S$終了通知を受けとるか，集合演算器から$R_S$オーバーフロー通知を受けとるかしたら，基本演算器に$R_S$送出準備を，DPNETと集合演算器に$R_T$受け取り準備を各々指示する。

⑥ $R_T$に対し，基本演算器，DPNETとも，②，③と同様の動作を行う。

⑦ 集合演算器は，$R_T$の1タップルを受けとるたびに，OP2演算を行い結果を出力する。結果はバッファに蓄えられ，まとめて中央論理装置，またはホスト計算機に送出される。

⑧ コントローラは，集合演算器から$R_T$の終了通知を受けとると，全$R_S$の読み込みが終了していればOP2演算を終了し次の演算を，そうでなければ，$R_S$の残りを読み込み②～④を行うことを，基本演算器，DPNET，集合演算器の各々に指示する。

以上が，システムの全体構成と処理の流れの概要である。OP1，OP3タイプの演算については，次章で述べる。

# 3　システム構成要素

## 3．1　基本演算器

図3－1に，基本演算器のブロック図を示す。図3－1に示すごとく，ここでは基本演算に必要な比較器の他に，送出済タップル判定ユニット（TDU）と，タップル転送許可信号バッファ（TSB）をもっている。

TSBは，次段から送られてくるタップル転送許可信号（TS）を蓄えておくためのバッファであり，次段でのタップル受け付け可能数を表示している。

TDUは，TSBが0になったとき（処理速度が読み込み速度に追い付かなくなったとき）セルの空読みを行い，次の回転で前回の続きのタップルを読んだり，$R_S$の読み込み中に集合演算器が一杯になり$R_S$の読み込みを中止した後，再び$R_S$を前回の続きから読み込んだりする（2．3節の④参照）のに必要なユニットである。

タップル送出ユニット（TSU）は，比較器からの比較結果信号（1），TDUからの未送出タップル信号（2），TSBからの転送許可信号（3）のAND条件が満されたら，TSU内のタップルを次段に送出する。ここでは，比較器での条件比較，TDUの処理，TSUへのタップル格納は並列に行なわれ，しかも，タップル長に比し，演算対象となる属性長は小さいので，上記判定を行っても，転送遅れが生じることはない。

## 3．2　DPNET

図2－2に示すごとく，DPNETは，アドレス割り付け器と，各ノードにタップル待ち行列用バッファ（TQB）をもったルーティングネットワークからなる。

図3－2にアドレス割り付け器を，図3－3

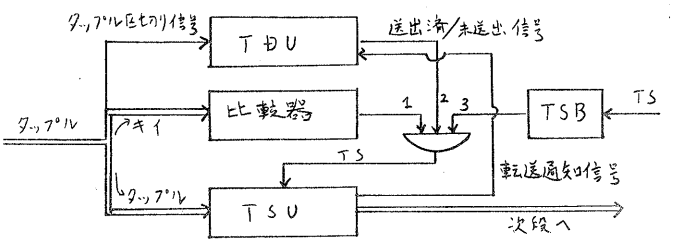



図3－1　基本演算器

にDPNETの各ノードを示す。ここで，TSB，TSUの役割は，基本演算器の場合と同じである。

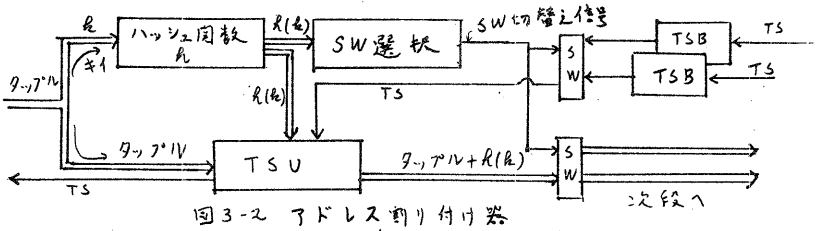



図3-2 アドレス割り付け器

図中，SW選択は，アドレス割り付け（ハッシュ関数）の結果を受けとり，タップルをどちらのノードに出力するかのスイッチ（SW）の切り替え信号の発出のみを行っている。TQB選択は，タップル格納数の多いほうのTQBからタップルを取り出すためのものである。

また，TQB，TSUでは，$R_S$用と$R_T$用の2種のバッファをもっている。これは，$R_S$の読み込み中に集合演算器が一杯になったときの，途中状態保存用である。

図2-2で，ルーティングネットワークの入口が，基本演算器の2倍（転送速度を2倍にすれば，物理的に2倍にする必要はない。）になっていることに注意されたい。これにより，DPNETの各ノードの，TQBの大きさを数タップル分とするだけで，途中でTQBがオーバーフローする（このとき前段のTSBが0になる。）確率を$10^{-3}$以下とすることができる。（4章参照）

## 3.3 集合演算器

ここでの集合演算器としては，次々に到着するタップルに遅れることなく，表2-1の集合演算が行なえることが望ましい。その意味で，ソーティングを行うもの[8][9]は，ソート用にデータを蓄える必要があるので，望ましくない。また，Systolic Array[7]などのように$R_S$，$R_T$の同時入力を行っているものは，本稿のDBMでは使えない。

そこで，図3-4に示す集合演算器を考案した。本集合演算器は，

① キイサーチプロセッサ（KSP）

② 演算プロセッサ（OP）

から構成されている。

KSPは，連想メモリとプロセッサで構成され，連想メモリには，連想メモリを節約する目的と，OP1タイプの演算を実現する目的のため，未登録キイのみをタップルの格納アドレス（$R_S$AD）とともに登録している。

OPは，タップル格納用メモリ（$R_S$M）とバッファ（$R_T$B）をもっており，この2つに格納されたタップルとKSPからの検索結果を使用して，集合演算を行っている。

$R_S$Mは，KSPに格納先アドレス（$R_S$AD）を知らせるためのアドレスカウンタ（AC）を持っている。同時にACは，$R_S$Mを節約するため，OPで不要なタップルは，KSPからのタップル不必要信号により，格納しないようにする（次に入力されたタップルを重ね書きする。）目的にも使用されている。

$R_T$Bも，$R_S$Mと同様の目的で，KPSからのタップル必要／不必要信号を受け取っている。

(1) KSP

KSPには，表3-1に示す5つの処理モー

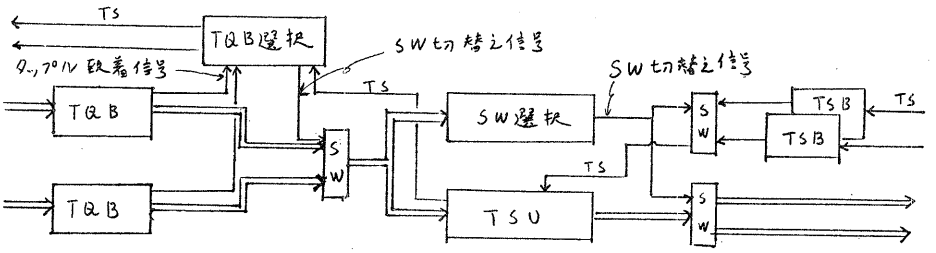



図3-3 DPNETのノード

Dがあり，キイ（演算対象となる属性値）とRsADを受けとり，キイのKSP内の連想メモリへの登録及び同一キイ検索を行っている。

KSP内では，キイは図3-5の形式で格納されている。ここでTP，TLは，RsMに格納された当該キイをもつタップルチェイン（チェインはOPが張る。）の，TOPとTAILアドレスである。

KSP内の処理で，検索の他に必要とされる処理は，最大2stepの格納処理と，キイの読みだし，検索結果の出力，キイの登録であるが，後の3つは並列に行うことができる。キイ検索も，LSI連想メモリ[12]を用いれば1回のメモリアクセスで，また並列ハッシュプロセッサ[13]を用いても，ほぼ1～2回のメモリアクセスで，検索を終えることができる。このことと，通常キイはタップルに比し短かいことを考えれば，KSPの処理時間をタップルのRTB，RsMへの格納時間と同程度とし，両者を並列に処理することは，充分可能である。

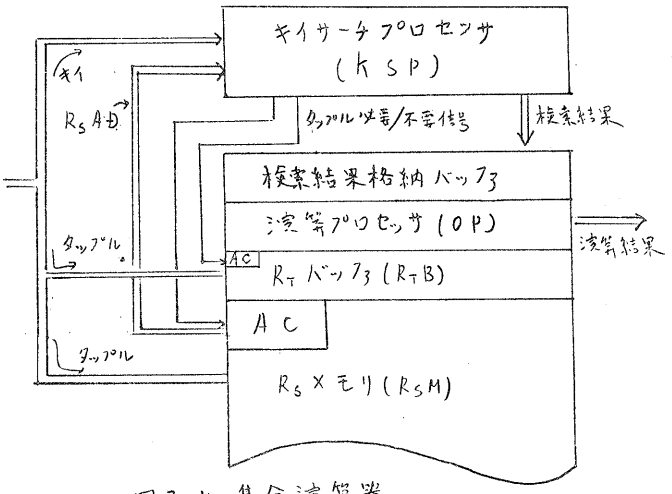



図3-4 集合演算器

（2）OP

OPは，KSPから受けとった検索結果とRsM，RTBに格納されているタップルから，次のいずれかの処理を行う。

① Rsタップルのチェイン（CHAIN）

KSPから受け取ったRsADとTLを用い，RsADを，TLで示されるタップルのチェインポインタ部に格納する。

② RTとRsの結合と出力（OUT(RT⌒Rs)）

RTBのタップルとRsMのタップルを結合し，演算（explicit JOIN）の結果として出力する。

表3-1 KSPの処理内容

| 処理モード | 入力 | 処理 | | | タップル必要信号 | | 検索結果の出力形式 |
|---|---|---|---|---|---|---|---|
| | | 検索 | キイ操作 | キイ登録 | RTB | RsM | |
| mode 1（無条件登録） | キイ RsAD | Y | RsAD → TL | △ | 不必要 | 必要 | キイ \| TP \| TL \| RsAD |
| | | N | RsAD → TP, TL | ○ | 不必要 | 必要 | キイ \| RsAD \| RsAD \| NULL |
| mode 2（新キイ登録） | キイ RsAD | Y | | X | 必要 | 不必要 | キイ \| TP \| TL \| Y |
| | | N | RsAD → TP, TL | ○ | 必要 | 必要 | キイ \| RsAD \| RsAD \| NULL |
| mode 3（同一キイ検索） | キイ | Y | | X | 必要 | 不必要 | キイ \| TP \| TL \| — |
| | | N | | X | 不必要 | 不必要 | |
| mode 4（新キイ検索） | キイ | Y | | X | 不必要 | 不必要 | |
| | | N | | X | 必要 | 不必要 | キイ \| TP \| TL \| — |
| mode 5（キイ有無検索） | キイ | Y | | X | 必要 | 不必要 | キイ \| TP \| TL \| Y |
| | | N | | X | 必要 | 不必要 | キイ \| TP \| TL \| N |

／；処理せず　Y；同一キイ有り　N；同一キイ無し　○；登録する　△；TLを変える　X；〃　せず

$R_S M$ のタップルがチェインされていれば，チェインされているタップルのすべてに対して，$R_T$タップルとの結合を行う。

出力時に，JOIN結果に対するRISTRICTIONがあれば，それも同時に行う。

③ $R_T$出力 (OUT($R_T$))

$R_T B$から取り出したタップルをそのまま出力する。

出力したタップルは，OP2タイプの演算では，それがそのまま演算結果となるが，OP1タイプの演算では，中間結果となり，全関係表入力終了後，中間結果を$R_S$として再入力する必要がある。

④ $R_S$出力 (OUT($R_S$))

$R_S M$に格納されている全タップルを演算結果として出力する。

⑤ AGGREGATION演算 (AG演算)

AG演算では，表3-2のごとく，$R_S$タップルに演算結果格納部を付加し(ACが格納部だけ余分にカウントする。)そこに演算結果を格納する。演算は，表3-2のごとく行うが，いずれも数stepsで終了する。

⑥ $R_T$タップル消去 (DEL($R_T$))

$R_T B$より，タップルを取り出すだけで何もしない。

以上が，OPの処理内容であるが，これらはKSPの処理とパイプライン的に結合され，並列処理される。

(3) 集合演算の実現法と状態遷移

集合演算の実現法を状態遷移図を使用し，図3-7 (OP1演算)，図3-8 (OP2演算)に示す。また，具体的演算に対し，KSP,OPが，各状態でどのような処理を行うかを，表3-3，表3-4に示す。

なお，OP3のDIVIDEは，implicit JOIN，PROJECTION，AG演算のCardinality Countを組み合せることにより実現できるので[14]，ここには示してない。

## 4. 評価と考察

本稿で提案したDBMは，まだ設計段階である。従ってここでは，ハッシュ関数により入力データの分布をランダムにできると仮定した場合の装置するハード量と期待される効果についての確率的評価を行った後，集合演算器の入力遅延性と$R_T B$の大きさについて考察する。

### 4.1 DPNETの各ノードの待ち行列数

DPNETの各ノードの待ち行列数分布を次の前提のもとに求める。

① 各ノードへは，T時間ごとにタップルが到着する。

② 各ノードは，T時間で1つのタップルを次の段に送ることができる。

すると，第一段目のノードでは，同時に2つのタップルが到着することがないので待ちが発生することはない。

| キイ | TP | TL |
|---|---|---|

図3-5 KSPでのキイ格納形式

| $R_S$タップル | AG演算用付加フィールド |
|---|---|

図3-6 AG演算時の$R_S$タップル格納形式

表3-2 AG演算処理

| 項番 | AG演算 | 追加フィールド名 | AG演算処理 | |
|---|---|---|---|---|
| | | | 新キイ* | 二度目からのキイ* |
| 1 | max | MAX | $A_x^{**} \rightarrow$MAX | if $A_x$>MAX then $A_x \rightarrow$MAX else ; |
| 2 | min | MIN | $A_x^{**} \rightarrow$MIN | if $A_x$<MIN then $A_x \rightarrow$MIN else ; |
| 3 | sum | SUM | $A_x^{**} \rightarrow$SUM | SUM+$A_x \rightarrow$SUM |
| 4 | cardinality count | COUNT | 1 $\rightarrow$COUNT | COUNT+1 $\rightarrow$COUNT |
| 5 | average | COUNT, SUM AVERAGE | 項番3，4を行った後 SUM/COUNT $\rightarrow$AVERAGE | |

*；新キイかどうかの判定は，表3-1のmode 2，mode 4の検索結果出力形式から判定する。

**$A_x$；$R_T B$に格納されたタップルの演算対象属性フィールド名

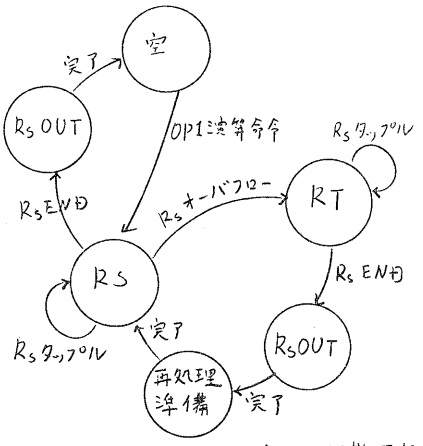



図3-7 OP1演算の状態遷移図

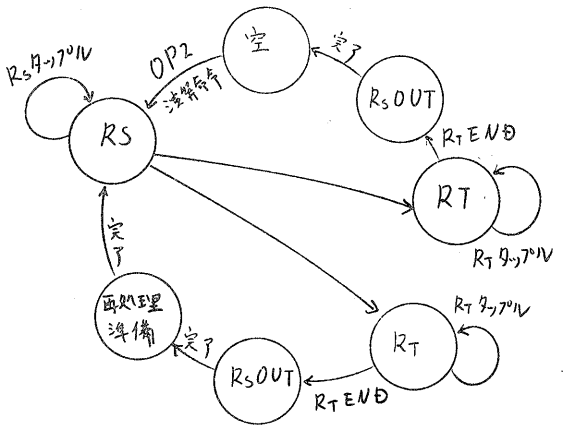



図3-8 OP2演算の状態遷移図

表3-3 OP1演算の状態と処理

| 状態 / 演算 | RS | | RT | | RS OUT | | 再処理準備 |
|---|---|---|---|---|---|---|---|
| | KSP | OP | KSP | OP | KSP | OP | |
| PROJECTION | mode 2 | DEL($R_T$) | mode 4 | OUT($R_T$) | | OUT($R_S$) | OUT($R_S$)として出力されたタップルを，$R_S$として再入力する。 |
| AGGREGATION | mode 2 | AG演算 | mode 5 | Y：AG演算<br>N：OUT($R_T$) | | OUT($R_S$) | |

表3-4 OP2演算の状態と処理

| 状態 / 演算 | RS | | RT | | RS OUT | | 再処理準備 |
|---|---|---|---|---|---|---|---|
| | KSP | OP | KSP | OP | KSP | OP | |
| explicit JOIN | mode 1 | CHAIN | mode 3 | OUT($\widehat{R_T R_S}$) | | | RSの残りを再入力する。 |
| implicit JOIN<br>INTERSECTION | mode 2 | DEL($R_T$) | mode 3 | OUT($R_T$) | | | |
| UNION | mode 2 | DEL($R_T$) | mode 4 | OUT($R_T$) | | OUT($R_S$) | |
| DIFFRENCE | mode 2 | DEL($R_T$) | mode 4 | OUT($R_T$) | | | |

第2段目のノードでは，

- ⓐ 2つ同時に到着する確率… $(1/4)^2$
- ⓑ 1つだけ到着する確率… $2\cdot(1/4)\cdot(3/4)$
- ⓒ 1つも到着しない確率… $(3/4)^2$

であるので，時刻 $t\cdot T$ における待ち行列数が $n$ である確率を $P_t(n)$ とおくと

$$\left.\begin{aligned} P_t(0) &= (15/16)\cdot P_{t-1}(0) + (9/16)\,P_{t-1}(1) \\ P_t(n) &= (6/16)\cdot P_{t-1}(n) + (9/16)\,P_{t-1}(n+1) \\ &\quad + (1/16)\,P_{t-1}(n-1) \qquad n \geq 1 \end{aligned}\right\} \quad 4-1$$

これより，$t\to\infty$ の平衡状態における待ち行列の分布関数 $P_\infty(n)$，待ち行列数の期待値 $E$ は，

$$P_{00}(n) = (6/9)(1/4)^n \quad , \quad E = 9/64$$

となる。

同様に，3段目以下の各ノードの到着確率も2段目と同じになることが示せる。

## 4．2　TQBオーバフロー確率

図3－3のTQBの容量を，$q/2$ タップル分（TQBは2つあるので，1ノード当り $q$ タップル分）とし，連続 $n$・T時間オーバーフローが発生しない確率を求める。 $n$・T時間後にオーバーフローが発生する確率は，4－1式で，$P_x(q+1)$ となる。これより，$0 \leq n \leq q-1$ では，4－1式をそのまま使用し，

◎ $n = q$ では

$$P_x(q) = (6/16)P_x(q) + (1/16)P_x(q-1)$$

◎ $n = q+1$ では

$$P_x(q+1) = P_{x-1}(q+1) + (1/16)P_{x-1}(q) \qquad \text{4-2}$$

として，$P_x(q+1)$ を求めると $P_x(q+1)$ は，$n$・T時間内に1回以上オーバーフローが発生する確率となる。これを計算した結果を図4－1に示す。

図より，RsMを $10^3$ タップル分，$R_T$ の大きさを $10^6$ タップル分としても，$q$ を各々6，8個と用意すれば，TQBのオーバーフロー確率を，$10^{-3}$（1000回の集合演算で1回の発生）程度とすることができる。

## 4．3　集合演算器の利用率

RS状態では，どれか1つの集合演算器が一杯になると，Rsタップルの入力を中止し，RT状態への切り替えを行っている。このとき，他の集合演算器にどの位のタップルが格納されているか（これを，ここでは利用率と呼ぶことにする。）を，次の仮定のもとに計算する。

① タップルは1つずつ到着する。

② 到着したタップルは，$n$ 個の演算器のどれか1つが，ランダムに選択されてそこに格納される。

③ $n$ 個の演算機のどれか1つが，$q$ 個のタップルを格納したら，入力を中止する。

$q$ 個の入力があったとき，特定の1つの演算器に $i$ 個の入力がある確率 $f(m, q, i)$ は2項分布となるので，最高 $n$ 個の入力があった時，他の演算器に合計 $i$ 個のタップルが入力されている確率 $P(m, n, i)$ は，次式のごとくなる。

◎ $i \leq n$

$$P(m,n,i) = f(m, n+i-1, n-1)\cdot n\cdot(1/n) = \binom{n+i-1}{n-1}\left(\frac{1}{m}\right)^{n-1}\left(\frac{n-1}{m}\right)^{i-1}$$

◎ $n \leq i \leq (n-1)\cdot(m-1)$

$$P(m,n,i) = f(m, n+i-1, n-1) \times \left\{1 - \sum_{j=0}^{i-n} P(m-1, n, j)\right\} \qquad \text{4-3}$$

これより，$i$ の期待値 $E_i$ と集合演算器の利用率 $E_U$ は，次式となる。

$$E_i = \sum_{i=0}^{(n-1)(m-1)} i\,P(m,n,i)$$

$$E_U = \{E_i/[(m-1)\cdot n]\}\cdot 100 \qquad \text{4-4}$$

図4－2に，$E_U$ の計算結果を示す。これより，$m$ を数十，$n$ を数百とすれば，80％程度の利用率が得られることが判る。

## 4．4　集合演算器の入力追随性とバッファサイズ

集合演算器の入力追随性が問題となるのは，RPSとOPの処理がパイプライン化されているRS，RT状態に於いてであるが，OPがOUT（$R_T \to R_S$）を行う時以外は，OPもしデータ転送十数 $\mu$sec）の処理しか実行していないので

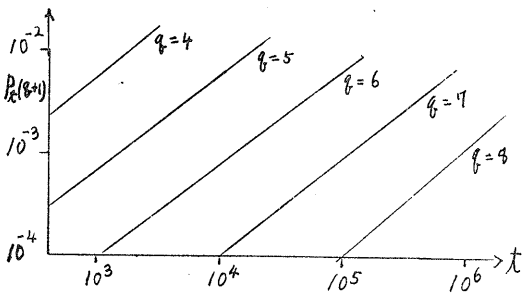



図4－1　TQBオーバーフロー確率

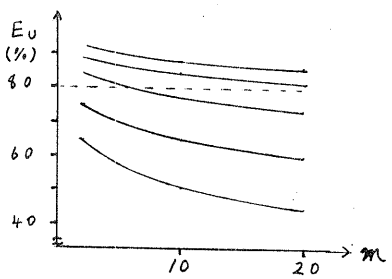



図4－2　集合演算器利用率

データの流れが止ることはない。

OUT($R_T \widehat{} R_S$)により，OPの処理が遅れると，$R_T$Bがオーバーフローし，DBPNETがデータを集合演算器に送れなくなる。このとき，3.1で述べたごとく，セル論理装置は1回転分の空読みを行う。この間OPは，$R_T$B，$R_S$Bのタップルに対しOUT($R_T \widehat{} R_S$)を行っている。従って，$R_T$Bの大きさを，空読み時間とOPの処理時間が一致する程度にしておけば，OUT($R_T \widehat{} R_S$)でも，無駄な遊びがOPに生じることはない。

$R_T$Bオーバフローが恒常的に発生するのは，タップルが到着する率とサービスが終了する率が等しいか，前者のちが大きくなった時である。OUT($R_S \widehat{} R_T$)の処理時間は，出力量に比例するので，これは，入力量の2倍以上の出力があるとき発生する。従って，$R_T$Bの大きさは，補助記憶の1セル分の記憶容量の1/2より大きくする必要はない。

## 5. あとがき

本稿では，セル論理装置の並列読み出し速度に応じてダイナミッククラスタリングを行うDBPNETと，タップルの転送速度に遅れることなく関係代数演算等を行う集合演算器を用いることにより，$O(n)/P$（$n$；タップル数，$P$；セル論理装置の並列読み出し数）で集合演算が行えるDBMを提案した。

補助記憶としてディスクを使用した場合，DBPNETの入口数は，32～64個でよい。また，各ノードに必要なバッファは10タップル分程度でよい。集合演算器を，$R_S$Mに64KByte，$R_T$Bに数KByte，HPSのキイ格納部に16KByte程度用意すれば充分である。また，ディスクからの転送速度を1M～1.5Mbyte/secとし，1タップル100バイトとすると，100μsec～66μsecで1タップル分の処理ができればよい。

残された問題点として，① explicit JOINの性格と$R_T$Bの大きさに関する詳細な解析，② ハッシュ関数の選択法，③ 集合演算器の数とデータの実現値の種類数や母程度が，後者のちが少ない場合の処理，④ 障害復旧法の検討，などがある。

これらについては，本稿で述べたDBMの設計の詳細化とともに，今後検討を進めて行く予定である。

〈謝辞〉 日頃御指導頂く第一研究室山下室長，春藤調査役をはじめ，種々御討論頂いた第一研究室の諸兄に深謝する。

〈参考文献〉

1. IE³ TRANSACTION on COMPUTERS VOL-28 NO.6 1979
2. IE³ COMPUTER VOL 12 NO.1? 1979.3
以上2つは，DBMの特集号である。中の個々の論文は省略する。
3. E. BABB Implementing a Relational Data base by means of specialized Hardware A.C.M. trans. on Data base Systems VOL 4. 1. 1979. 3 PP. 1～24
4. 青木他 動的マークビットによるリレーショナル演算の並列処理とその評価 信学会 計算機研究会 EC79-66
5. S. Umamura et al. the Desing and implemontation of a Magnetic-Bubble Data Base Machine Proc IFIP 80
6. B.W. Wah et al DIALOG: A distributed process organization for data base machine Proc. AFIPS NCC 1980 P243
7. H.T. Kung et al Systolic (VLSI) Arrays for Relational Data base Operation Proc ACM SIGMOD 1980 P105
8. 上尾 ソーテングハードウエアを用いたデータベースマシン 京大 大型計算機センター 第1回セミナー報告 1980.3
9. Y. Tanaka Pipeline Searching and Sorting Modules as Components of Data Flow Data base Computers Proc. IFIP 80
10. M. Maekawa Quick Parallel JOIN and Sorting ALGORITHMS 13th. IBM Computerscience Symp. 1979.10 P III-11
11. D.E. Shaw A Relational Data base Machine Architecture 5th workshop on computer Architecture for nonnumeric processing ACM SIGARCH, SIGMOD, SIGIR, P84
12. 小倉他 1Kビット連想メモリLSI 信学会 EC80-44
13. E. Goto et al Parallel Hashing Algorithms Technical Rep. of Tokyo Univ. TR-76-16
14. 古川 データベースの知的アクセスに関する研究 電総研 研究報告 第804号
15. 植村，前川 著 データベースマシン 情報処理叢書1

盛光印刷所