解　説

# ファイル・プロセッサとデータベース・マシン*

前　川　　守**

## 1. はじめに

近年専用プロセッサの1つとして，データベース・マシン及びファイル・プロセッサなるものが注目をあび，広くその存在が認識されるようになってきた．これは従来必ずしも充分に考慮を払われていなかった非数値処理を一般計算から分離し，それに適したアーキラクチャを有するプロセッサを構成しようとするものである．日本でも大学，研究所を中心として，データベース・マシンの研究が注目をあびている[34),37),39)]．データベース・マシンの解説も既にいくつか出されており[36),41)]，特に本誌では，関野，植村両氏による優れた解説がある[41)]．この上，更に解説を重ねるのは屋上屋を架す感がある．しかし，非数値処理の重要性に鑑み，敢えて屋上に屋を架してみる次第である．なお，この解説は関野，植村両氏の解説の存在を前提としており，本稿と共に関野，植村両氏の解説[41)]を併読されたい．

さて，この解説の目的は2つである．1はデータベース・マシンのアーキテクチャを分類し，各方式の利点，欠点を明らかにすることである．さらには，実際のシステム（提案中のものを含めて）がどの方式に属するかを分類し，どのような工夫が施されているかを解説する．もう1つの目的はデータベース・マシンの大きな柱となっている連想処理（Associative Processing）方式の得失を明らかにすることである．

データベース・マシンおよびファイル・プロセッサを考える場合，アーキテクチャ的には，2つに分けて考えることができる．1つは専用プロセッサとしてのデータベース・マシンまたはファイル・プロセッサを取り囲む全体システムのアーキテクチャであり，他は専用プロセッサ自身のアーキテクチャである．システム・アーキテクチャについては**2.**で述べることとし，**3.**, **4.**ではデータベース・マシンおよびファイル・プロセッサの内部アーキテクチャについて述べることとする．**5.**ではデータベース・マシンの大きな柱の1つとなっている連想処理（Associative Processing）方式の定量的評価を行い，連想処理方式の普及の条件を明らかにする．

## 2. システム・アーキテクチャ

データベース・マシンおよびファイル・プロセッサは非数値処理を専門に行うプロセッサである．これらのプロセッサが全体システムの中でどのように結合されるかの方式を，システム・アーキテクチャと呼ぶこととする．システム・アーキテクチャを定める最も重要な点はメモリとロジックとの結合の様式である．ロジックとして数値処理を中心としたロジックと非数値処理を中心としたロジックに分離し，前者を Central Processing Unit (CPU)，後者を Non-Numeric Processing Unit (NNP) で処理させることとする．入出力のデータ転送のみを司どるプロセッサが必要な場合は Input/Output Processor (IOP) と呼ぶこととする．メモリとしては CPU のメモリ (MM)，NNP の主としてプログラム格納のためのメモリ (NM)，二次記憶装置 (SM)，そして，バックアップ・メモリ (BM) とに分離して考えることとする．これらのメモリが実際にどのようなデバイスで実現されるかはシステム・アーキテクチャ上大きな問題ではないが，SM としてはディスク，CCD，磁気バブル等が考えられている．さて，これらのロジックおよびメモリの結合の様式により次のような分類ができる．

a. 集中ロジック
b. 分散ロジック
c. ネットワーク

### 2.1 集中ロジック

CPU と NNP を集中化した方式である．CPU と NNP の結合の方式により**図-1**（a），(b)，(c)(次頁参照)の3方式に分類できる．方式（a）は NNP が

---

* File Processors and Data Base Machines by Mamoru MAEKAWA (Toshiba Research and Development Center).

** 東京芝浦電気(株)総合研究所

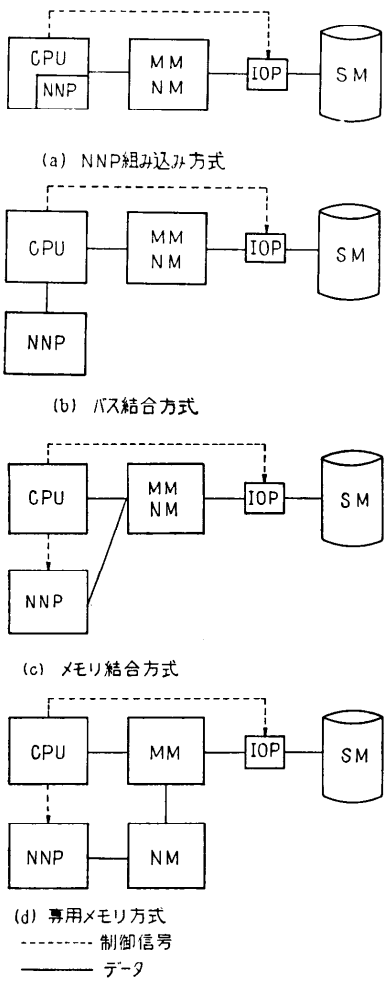



**図-1** 集中ロジック

CPU に組み込まれた方式であり，方式(b)は NNP が CPU にバス結合されている方式である．最近の非数値処理にも適したプロセッサは，このいずれかの範ちゅうに入ると思われる．方式(c)はメモリ結合による付加プロセッサである．この方式のファイル・プロセッサとしては IFAM[9], APCS[17], LEECH[21] 等々[40] いくつかの提案および実験がある．それらの中，代表的なものについては **4.** で述べる．(d)の方式は専用メモリ（通常非数値処理用の特殊メモリ）を有する付加プロセッサを用いる方式で，例えば Intelligent Memory[13] をシステムに組み込む場合には，この方式になると考えられる．集中ロジック方式においては，あくまでも CPU が主であり，NNP および IOP は従プロセッサとして動作する．集中ロジック方式の

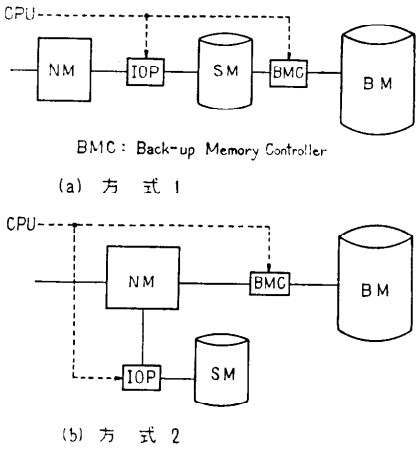


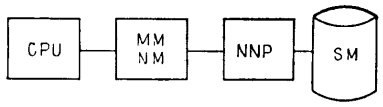

**図-2** バックアップ・メモリ

利点，欠点については **4.** で述べる．

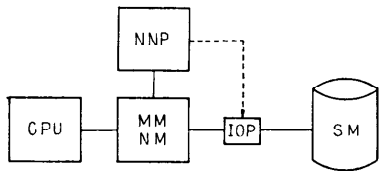
バックアップ・メモリ (BM) は図-1 に記入されていないが，その結合の方式としては**図-2** に示すように SM 接続される方式と NM (MM) に直接結合される方式とがある．

### 2.2 分散ロジック

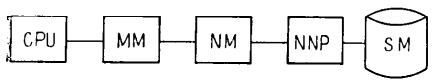
非数値処理専用ロジック，NNP を CPU とは独立に動作させる方式である．NM を MM から分離するか否か，IOP を NNP に含むか否か等により**図-3**(次頁参照) に示すような5つの方式に分類できる．CPU と NNP の間には通常何らかの信号をやりとりできる信号線が設けられる．(a)および(c)の方式は NNP が直接 SM 上または SM からのデータに操作を加えられるようにした方式であり，広い意味でのインテリジェント・ディスク (Intelligent Disk, 知能ディスク) 方式と考えることができる．この分類に属するシステムとしては ECAM[2], IMSAI 108[15], RARES[16], SETF[24], RAP[25] 等がある．IMSAI 108 は文字通り Intelligent Disk と銘うたれているが，他のシステムは Intelligent Disk であるよりも連想処理 (Associative Processing) 方式であることの方がその特徴として大きい．これらシステムの内部アーキテクチャについては次章で述べる．(b)のマルチプロセッサ方式は大きなリアル・タイム・システムやポリプロセッサ方式でよく採用される方式である[42]．専用化の手段としてはファームウェアもよく用いられる．(d)のバックエンド・コンピュータ方式は XDMS[6] により，よく知られるようになった．(e)の方式は NM を SM に移すことにより(a)と(d)の両方式を兼ね備えるようにした方式である．CASSM[7] がこの方式をとってい



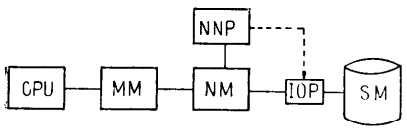

(a) インテリジェント・ディスク方式 1



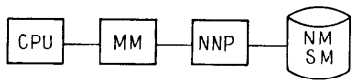

(b) マルチプロセッサ方式




(c) インテリジェント・ディスク方式 2




(d) バックエンド・コンピュータ方式

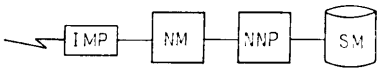



(e) スタンド・アロン・インテリジェント・ディスク方式

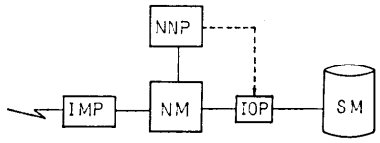
**図-3** 非数値処理専用ロジック

る．なお分散ロジック方式の利点，欠点については解説[41]を参照されたい．

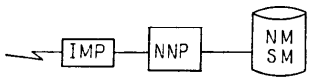
分散方式においてバックアップメモリ (BM) が必要な場合には接続する相手方のメモリの種類により3つの方式に分類できる．1つは MM に接続する方式であり，他は NM, SM にそれぞれ接続する方式である．RAP では SM に接続する方式を提案している[30]．SM の容量が小さく BM が容量的には通常の SM の代りを果す場合も考えられる．その場合には MM 結合方式は集中ロジック方式に属すると考えられるが SM 結合方式では二次記憶装置と主記憶装置とを結合するチャネルにメモリとロジックが装備されたと考えることもできる．この方式はチャネル方式として有沢氏らにより提唱されている[34]．チャネル方式ではチャネルは必要なデータのみを通過させる一種のフィルターとして働く．




(a) インテリジェント・ディスク方式




(b) データ・コンピュータ方式




(c) スタンド・アロン・インテリジェント・ディスク方式

IMP: Interface Message Processor

**図-4** ネットワーク

### 2.3 ネットワーク

コンピュータ・ネットワークの構築法としては，大きく分けて2つの方法がある．1つは既存のコンピュータ・システムを結合していく方式で，データベース・マシンとして特に問題はない．他は機能分散型のネットワークを構成する場合である．この方式では，CPU および MM がネットワークにより置き替えられる．そのためには MM と NM が分離されていることが必要であり，従って図-3 の (c), (d), (e) の方式において CPU と MM をネットワークで置き替えた方式が考えられる．**図-4** にその方式を示す．ネットワーク上のデータベースの諸問題については解説[43]を参照されたい．

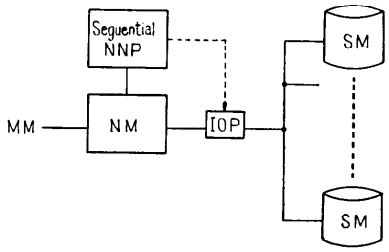
## 3. データベース・マシン・アーキテクチャ

分散ロジック方式において，より高性能を求めるための手段として，次の2つが考えられている．

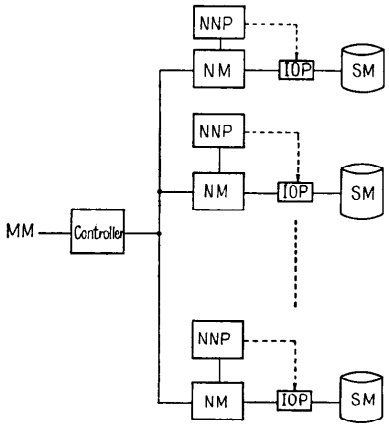
a. メイン・メモリ (MM) と二次記憶 (SM) とのデータ転送量を少なくするため，できるだけデータの存在するところ，すなわち，SM 上または SM に近いところでもって処理を行う．

b. 連想処理を導入する．

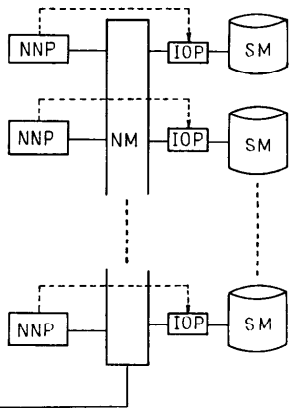
これら手段の実現法として，データベース・マシンのアーキテクチャに関し，いくつもの提案，実験が行われている．これらは**図-5**(次頁参照)に示すように分類することができる．方式(a)のバックエンド・コンピュータ方式は NM と SM の間で従来と同様の問題を有すると考えられ，本質的には，上記手段のいずれ



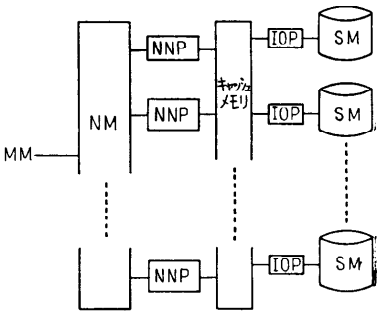

(a) バックエンド・コンピュータ方式



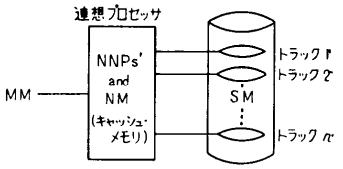

(b) インテリジェント・ディスク方式 (バス結合)



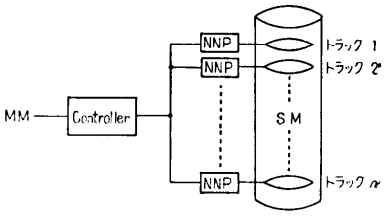

(c) インテリジェント・ディスク方式 (メモリ結合)



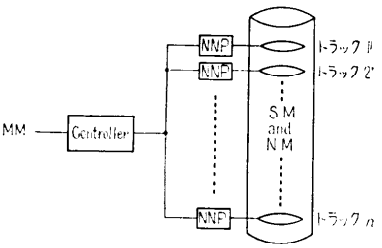

(d) インテリジェント・ディスク方式 (キャッシュ・メモリ)




(e) 連想処理方式 (キャッシュ・メモリ)




(f) Logic-per-track方式 (1)




(g) Logic-per-track方式 (2)

**図-5** データベース・マシン・アーキテクチャ

をも適用していないと考えられる．方式(b)～(d)は手段(a)を限定的に適用した方式である．方式(e)は更に連想処理を導入した方式である．方式(f),(g)は更に1歩を進め，データを直接SM上で処理するようにすると共に，トラック単位の連想処理を導入した方式である．方式(a)としてはXDMS[5]，方式(d)としてはIMSAI 108[14]，方式(e)としてはSETF[23]，方式(f)としてはECAM[2]，RAP[24]，その変形としてRARES[15]，方式(g)としてはCASSM[31]がある．

さて連想処理方式の利点としては次の項目をあげることができる[3),9)]．

a. 高性能．

b. メモリの使用効率が高い.
c. データ構造の変化の影響を受けることが少ない.
d. 検索条件の変化に対して比較的容易に対処できる.
e. 通常ソフトウェアが簡単になる.

一方欠点としては，次の項目をあげることができる.

a. 高価格.
b. 連想メモリと他のメモリまたはデバイスとの間に大量のデータ転送が必要となる.
c. 連想処理を行うために，しばしば固定のデータ・フォーマットが要求される.
d. 連想メモリの大きさは現在限られている.
e. 連想処理は広く用いられていないため，必要なソフトウェアやアルゴリズムの開発が充分でない.

連想処理方式とシークェンシャル処理方式との性能比較は必ずしも容易ではないが，RAP ではトラックの数が200で検索条件にマッチするレコードの数が全レコード数の2%程度であるという想定のもとに，単純検索において100倍前後の性能向上が期待されている[26].

ECAM[2] は CCD メモリを用いた $10^9$ ビット（約100 M バイト）の容量を有する連想処理方式のデータベース・マシンであり実処理において現存の大型機の200倍程度の処理能力を有することが期待されている.

ECAM，RAP は二次記憶上の連想処理であるが，メイン・メモリ・レベルでの連想処理を行ったものとして，STARAN を用いた APCS では IBM 370/145 に対し単純検索において32〜110倍の性能向上が得られたことが報告されている[17]. また Intelligent Memory[13] では通常の RAM による方式に比べ 1,000 倍以上の性能向上が期待できるとしている. 連想処理方式の定量的評価は **5.** で行う.

さて，次にデータベース・マシンのアーキテクチャ上問題となるいくつかの重要な点について，各システムの解決法を述べる. 問題点として，次の項目を考える.

a. 検索法
b. 連想処理の制御
c. 連想メモリからのデータ出力の制御
d. バックアップ・メモリ

### 3.1 検索法

Logic-per-track 方式のシステムとしては，ECAM[2], RAP[25], CASSM[32], RARES[16], Parhami[27], Healy[14], 電総研[37]等があるが，それらの検索法の違いは，デリミタ（レコードまたはドメインを区切る印）の検出，複数ドメイン (Multidomain) の検索，レコードの出力等が1回のディスクの回転で行えるか否かの点にある. Healy と Parhami では1回転の間には1ドメインしか検索を行うことができないが，マッチングのとれたレコードには印をつけるマーキングの機能がある. このマーキングの機能により複数ドメインの検索を CPU の介入なしに何回かのディスク回転にわたって行うことができる. RAP と CASSM では複数ドメインの検索をディスク1回転で行える. 電総研の提案になるデータベース・マシンでは磁気バブルを用いており，ディスクと異なり回転型メモリでないため複数ドメイン，複数キーによる探索がロジックのスピードによる制約を考えることなく行えることが期待されている. CASSM では検索されたレコードには通常印がつけられ，次の回転で出力されるが，RAP では検索されたレコードは通常直ちに出力される. 従って RAP の方が高性能を得やすいと考えられるが，CASSM ではこれを補なうために，検索と出力をパイプライン的に行い実効的に RAP と同程度の性能を得られるようにしている. RAP, CASSM ではレコード検索機能に加えて，レコードの挿入，削除，修正の機能，ガーベージ・コレクションの機能がハードウェアで実現されており，CPU の介入なしに自動的に行われる.

### 3.2 連想処理の制御

図-5 において方式（e）〜（g）は連想処理を積極的に取り入れた方式である. これらの方式に属するシステムとしては既に述べたように，ECAM[2], SETF[24], RAP[25], RARES[16], CASSM[32], 電総研[37] がある. ECAM, SETF を除いてはいずれも，現段階ではセルの数は数個か，または提案のレベルにとどまっており，その意味では，連想処理方式の実現性を立証しているとは言い難い段階である. SETF では商用の連想処理プロセッサ STARAN を用いて 64 個のトラックに対して連想処理を施しており，それなりの実現性を立証しているものと考えられるが，処理の方式は他のシステムに比べずっと簡単である. 現在の数百メガバイト程度のディスクに対してトラック単位の完全な連想処理を施すことを考えると，プロセッサの数は1万を越すと考えられその制御は容易ではないと思われ

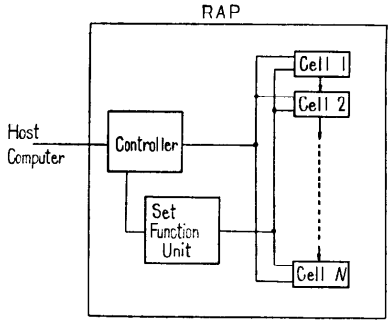



図-6 RAP アーキテクチャ

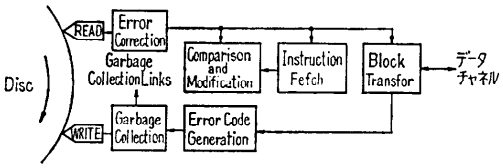



図-7 CASSM セル・アーキテクチャ

る．ECAM は現在製作中であり，完成すれば連想処理方式による最初の本格的データベース・マシンになると思われる．容量は $10^9$ ビットと大きい．

**図-6** にRAPの全体アーキテクチャを示す．Controller が全体の制御に当たり，Set Function Unit は全セルに対する総和，最大，最小，平均等の計算を行うのに用いられる．CASSM も同様のアーキテクチャを有するが，CASSM の命令体系は RAP に比べて低レベルであり RAP の有するような Set Function Unit を有しない．CASSM の各セルの構造を**図-7** に示す．

### 3.3 データ出力の制御

連想処理における問題の1つは，並列処理されて出てくる結果をいかに出力するかということである 検索の条件によっては多数のレコードが同時に出力される必要がある．STARAN[29] を用いた SETF[24] では STARAN の連想メモリが直列並列変換器 (Serial/Parallel Converter) として用いられており，メイン・メモリには，直列にデータを送っている．RAP[25]，CASSM[32] では同時には1個のレコードしか送り出せないようになっており，残ったレコードには二次記憶上に印をつけておき，次の回転で取り出しを行うようにしている．この方式は簡単だが出力に時間がかかる．これに対し RARES[16] では，SETF と RAP，CASSM 方式の中間の方式を提唱している．**図-8** に RARES の構成を示す RARES では各レコードをトラックに直交する方向に格納する．従って各レコードの各バイトは並列に出力される (RARES ではバイト単位に並列だが，各バイト内では直列)．この出力を直列並列変換を行うところは SETF と同じ方式である．一方，ディスクはいくつかのバンドに分割されており (各バンドから原則として1個のレコードが出力される)，そのため各バンドからの結果は同時には1個しか出力できないように制限されている．この点では RAP, RARES と同じ方式である．しかし，バンドの数はトラックの数よりずっと少ないため衝突の起こる回数も少なくなり，従って迅速な出力が可能である．

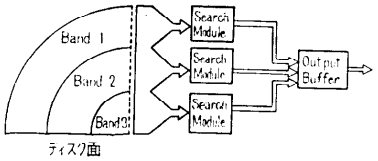



図-8 RARES アーキテクチャ

### 3.4 バックアップ・メモリ

二次記憶上の連想処理方式での問題の1つは，これら連想メモリの容量を充分大きくとることができないことである．この対策として，RAP では仮想メモリ (Virtual Memory) 化を考慮している．RAP の方式は RAP の各セルを2個のトラックを組としてダブル・バッファリング的に用い，RAP のトラックとバック・アップメモリの間で直接データ転送を行う方式である．これらの制御は，バックエンド・コンピュータでもって行う．

## 4. ファイル・プロセッサ

非数値処理のために提案または実験されている特殊プロセッサとしては，既に述べたように APCS[17]，IFAM[9]，LEECH[21]，マージ・プロセッサ[36]，Intelligent Memory[13] 等がある．本章では CPU の付加プロセッサ形式をとる特殊プロセッサの代表として LEECH を取り上げ解説する．LEECH のような特殊プロセッサ方式は従来のシークェンシャル方式より次のような理由で優れているという．

a. LEECH は非数値処理に特に向くように設計されているので高性能である．

b. LEECH は命令列を記憶できるので，命令参照のためのメイン・メモリへの負担がずっと減少する．すなわち，データ処理効率が高い．

c. データに対してはやはり少量だが高速メモリを内蔵しており，その意味でもメイン・メモリへの

負担が減少する

一方 Logic-per-track 方式に対しては次の点で優れているという．

a. Logic-per-track 方式はディスク等の回転型メモリと一体化した形で実現されるためマージ等の複数のデータ・ストリームを必要とする操作を行うことができない．

b. ディスクを用いた場合はデータが一定のスピードで送られてくるため，操作はそのスピード内で処理できるものに限られている．

c. 二次記憶の性質に左右されることがない．

LEECH は次のような命令を備えている

a. 検索：かなり複雑な条件式による検索が可能である．

b. ソート

c. マージ

d. 結合 (Join)

e. 冗長レコードの削除

これらの操作は主としてファームウェアで実現されるが，そのアルゴリズムの特徴は操作を2段階に分けて性能向上を計っていることである．たとえば結合操作においては，まず第1段階としてハッシング (Hashing) を利用して，まず対象となるレコードの数を少なくし，それから第2段階で厳密に結合操作を行う．**図-9** において2つのリレーション A, B があり，その結合操作を行うことを考える．A1, B1 はかなり長いビット・リストである．まず第1段階として，リレーション A にハッシングを施し，1つでも対応するレコードがあるビットは1にする（ビット・リストは最初0），次にリレーション B に同じハッシングを施し，得られた値をビット・リスト A1 と比較し，もしその値が1ならば，そのレコードを必要として残し，0の場合は不必要として第2段階の操作から除外する．このリレーション B のハッシングにおいて必要とみなされたレコードに対してはビット・リスト B1 の対応するビットを1にする．このあと再びリレーション A にハッシングを施し，同様に不必要なレコードを除外する．これにより，かなりの数のレコードを除外することができる．第2段階としては残ったレコードに対して厳密な結合操作を施すことで操作は終了する．

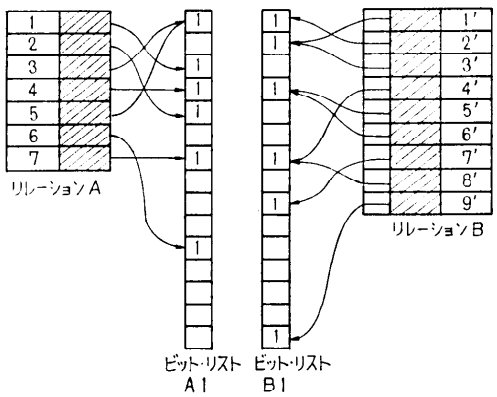



**図-9** LEECH の結合操作

## 5. 連想処理の定量的評価

連想処理方式の定量的評価を行うため**図-10** に示すモデルを考える．メモリは $n$ 個のブロックに分割されブロック内ではシークェンシャル・サーチが行われる．シークェンシャル・サーチではデータはプロセッサのメモリに転送され検索が行われる．連想処理方式では各メモリ・ブロックごとにロジックがあり並列に検索が行われる．コントローラは全体の制御を行う．変数を次のように定義する．

$P$：シークェンシャル処理におけるプロセッサの価格

$M$：シークェンシャル処理におけるプロセッサのメモリの価格

$m$：メモリの1ブロックの価格

$C$：連想処理におけるコントローラの価格

$p$：連想処理における各メモリ・ブロックのロジックの価格

$T$：各ブロック内でのレコードの検索時間

$N$：ブロックの数

次の3種の方式を比較することとする．

(a) リニア・サーチ：レコードはメモリを順次調べていくことにより検索される．この場合の平均検索時間は $(NT)/2$ である．検索にはプロセッサの処理も相当含まれると考えられる実効的な検索時間 $S_L$ として上記時間の倍 $NT$ を

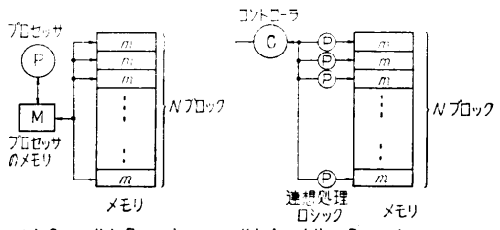



**図-10** Associative Processing Sequantial Processing

とる．

$$S_L=NT.$$

全システムの価格 $U_L$ は

$$U_L=P+M+mN$$

である．性能として検索時間の逆数をとると価格性能比は

$$E_L=\frac{1}{NT(p+mN)}$$

となる．

(b) トリー・サーチ：データ・ベースは木構造に構成されており，検索はインデックスをたどることにより行われる．この場合の平均検索時間 $S_H$ はプロセッサによる処理が同程度含まれると仮定すると

$$S_H=2T\log_B N$$

である．対数の底$B$の値としては現在のディスク等のメモリとプロセッサのスピード比により最適値が定まり大体100前後となる．後の数値例では $B=100$ と仮定する．全システムの価格はリニア・サーチとほぼ同じであり，同一と仮定すると

$$U_H=P+M+mN$$

となる．価格性能比 $E_H$ は

$$E_H=\frac{1}{2T(P+M+mN)\log_B N}$$

となる．

(c) 連想処理：検索時間は原則として一定である．

$$S_A=T.$$

価格 $U_A$ は

$$U_A=C+(p+m)N$$

である．価格性能比 $E_A$ は

$$E_A=\frac{1}{T\{C+(p+m)N\}}$$

となる．

次に各方式の相対的な性能価格比を比較するために次の2つの比

$$R_L=\frac{E_A}{E_L},\quad R_H=\frac{E_A}{E_H}$$

を考える．$R_L$ はリニア・サーチに対する連想処理方式の価格性能比の良さを示し，$R_H$ はトリー・サーチに対する場合である．$R_L$, $R_H$ は前述の式から次のように求められる．

$$R_L=\frac{(P+M+mN)N}{C+(p+m)N},\quad R_H=\frac{2(P+M+mN)\log_B N}{C+(p+m)N}.$$

さてここで注目すべきことは $R_L$ では分子が $N$ の2次式であるのに対し分母は $N$ の1次式である．すなわち $R_L$ は $N$ の1次のオーダである．これは $N$ の増加に対し，連想処理方式の価格性能比は $N$ に比例して上昇することを示す．すなわちプロセッサ（ここではロジック）を $N$ 台ならべれば性能は $N^2$ 倍になるのである[18),38)]．これが非数値処理の研究を有望なものとする1つの理由である．トリー・サーチに対しても $R_H$ は $N$ の増加関数であり（少なくとも大きな $N$ に対して）連想処理方式が優れていることを示している．次に数値例を示そう．

a) ロジックが比較的高価な場合．次の価格を想定する．

$P+M=C=8,000,000$ 円

$m=1,000$ 円

$p=50,000$

$B=100$（トリー・サーチにおける対数の底の値）

b) ロジックが比較的安価な場合．

$P, M, C, m, B$ については（a）と同じだが $p$ として100 円を想定する．

**表-1, 2** に $N$ を変数として考えた場合の $R_L$, $R_H$, $U_L, U_H, U_A$ を示す．これから次のような観察ができる．

a) ロジックの価格が低い場合には連想処理が価格性能比で有利であり，絶対価格でもほぼ変らない．

b) ロジックの価格が高い場合にはブロック数100,000 位のデータベースで価格性能比が最小となりかなり1を下回る．すなわち連想処理がかなり不利であり，しかも価格も非現実的な値とな

**表-1** $p=50,000$ 円の場合

| ブロック数 / 指標 | 100 | 1,000 | 10,000 | 100,000 | 1,000,000 | 10,000,000 |
|---|---|---|---|---|---|---|
| $R_L$ | 61.8 | 152.5 | 347.5 | 2,114.3 | 19,761.6 | 196,232.2 |
| $R_H$ | 1.24 | 0.46 | 0.14 | 0.11 | 0.12 | 0.14 |
| $U_L, U_H$ (単位百万円) | 8.1 | 9.0 | 18.0 | 108.0 | 1,008.0 | 10,008.0 |
| $U_A$ (単位百万円) | 13.1 | 59.0 | 518.0 | 5,108.0 | 51,008.0 | 510,008.0 |

**表-2** $p=100$ 円の場合

| ブロック数 / 指標 | 100 | 1,000 | 10,000 | 100,000 | 1,000,000 | 10,000,000 |
|---|---|---|---|---|---|---|
| $R_L$ | 99.9 | 989.0 | 9,473.7 | 91,525.4 | 909,747.3 | 9,091,570.0 |
| $R_H$ | 2.00 | 2.97 | 3.79 | 4.58 | 5.46 | 6.36 |
| $U_L, U_H$ (単位百万円) | 8.1 | 9.0 | 18.0 | 108.0 | 1,008.0 | 10,008.0 |
| $U_A$ (単位百万円) | 8.1 | 9.1 | 19.0 | 118.0 | 1,108.0 | 11,008.0 |

る.

連想処理方式の他の利点の1つとしては検索において条件に適合するレコードの数が多数個ある場合には，連想処理方式の相対的性能向上が著しくなることである．これは連想処理方式においては検索は原則的に加算的であるのに比し，シークェンシャル方式では乗算的であるからである[10]．前述の RAP の性能予測においては全レコードの2%程度 (2,000 レコード) のレコードがマッチするものと仮定しており，これが 100 倍前後も性能が向上している主原因となっている しかしこの利点も結果の出力が所栓シークェンシャルにならざるを得ないために，理想的な場合でも1ブロック当りのレコードの数をめどにおさえられる．連想処理方式の不利な点としてはデータ処理の特殊性及び常に $N$ ブロックを満たすファイルが存在するものでないことによる使用効率の低下があり，この利点，欠点がほぼ通常相殺されるものと予測される．これらのことを勘案して，ロジックの価格がメモリの1割程度に達したときは連想処理が有利であり，そこに達するまで（現在も含めて）は連想処理は不利であり特殊な場合を除いては普及は難しいと判断される．

## 6. おわりに

専用プロセッサとしてのデータベース・マシンまたはファイル・プロセッサのアーキテクチャを分類し，対応するシステムについて述べた．これら専用プロセッサの将来動向を考える場合図-1 の方式(a), (d)のような付加プロセッサの方式の実現性は比較的高いと考えられる．

それは1つには Logic-per-track 方式に比べ低価格であり，また汎用性が高いことである．普及の条件としては，低価格の連想プロセッサまたは連想メモリの出現が必要であろう．特殊な応用分野，たとえば実時間処理システムのように性能が決定的要素であるシステムでは，価格にかかわらず用いられる公算は大きい．これは連想プロセッサが米国では既に軍用には応用されていることからも察せられる．

データベース・マシンのアーキテクチャ (図-5) においては，方式(a)のバックエンド・コンピュータ方式は，個々の応用システムの中で適宜用いられていくと考えられる．特に最近のミニコン，マイクロ・コンピュータの発達に伴い，それらをデータベース処理に応用したシステムは多かれ少なかれ，バックエンド・コンピュータ方式になると考えられる．この方式の最大の問題はホスト計算機とのインタフェースであり，特に言語体系が問題である．方式(b)～(d)のインテリジェント・ディスク方式は大きな分散型システムで用いられていくものと思われる．技術的にはマルチプロセッサ方式とほぼ同様の問題を有する．ホスト計算機とのインタフェースはバックエンド・コンピュータ方式と同様，大きな問題である．方式(f),(g)の Logic-per-track 方式については前章で定量的評価を原理面から述べたが，普及の条件として次の3つを考える必要がある．

a) 絶対価格
b) 価格性能比
c) 絶対的性能

前章で述べたようにロジックが 50,000 円位の場合には連想処理方式は絶対価格で非現実的な値となり，価格性能比においても劣る．従って普及は難しいと考えられる．

ロジックが100円位（メモリ価格の1割位）の場合には連想処理は価格性能比で優れ，絶体価格でも大きな差がなく連想処理がかなり有利といえる．従ってロジックがメモリの1割位になる場合には大きな普及が望めると考えられる．

つぎにデバイス面から眺めた場合 Head-per-track ディスクはビット当りの価格が高く[35]，また小容量である．その面から考える限り少数の実験システムを除いてはディスクを用いての連想メモリは長期的には主流になるのは困難と考えられる．CCD または磁気バブルを用いての連想処理の実現性はこれらデバイスの発展に大きく左右されるが，これらデバイスを用いての連想メモリの低価格化が実現した時点で初めて連想処理方式のデータベース・マシンの広い普及が望めると考えられる．その過程においては，特殊用途に対しては経済性を無視して使用される場合ももちろん考えられる．

謝辞 この研究は工業技術院「パタン情報処理プロジェクト」の一環として行われた 5. の連想処理の定量的評価の部分は東芝総合研究所 石井 暁氏の援助をいただいた．

## 参考文献

1) D. R. Anderson: Data Base Technology, AFIPS Conf. Proc., Vol. 45, pp. 811～818 (1976).
2) G. A. Anderson and R. Y. Kain: A content-addressed memory designed for data base applicatios, Proc. 1976 International Conference on Parallel Processing, Wayne State University,

pp. 83～91 (Aug. 24～27, 1976).
3) R. I. Baum: The Architectual Design of a Secure Data Base Management System, Compcon Spring 76, pp. 113～117 (1976).
4) P. B. Berra: Some Problems in Associative Processor Applications to Data Base Management, AFIPS Conf. Proc., Vol. 43, pp. 1～5 (1974).
5) J. A. Bush, et al.: Some Implementations of Segment Sequential Functions, Third Annual Symposium on Computer Architecture, pp. 178～185 (1976).
6) R. H. Canaday et al.: A Back-end Computer for Data Base Management, Comm, ACM, Vol. 17, No. 10, pp. 575～582 (1974).
7) G. P. Copeland et al.: The Architecture of CASSM: A Cellular System for Non-Numeric Processing, First Annual Symposium on Computer Architecture, pp. 121～128 (1973).
8) G. F. Coulouris et al.: Toward Content-Addressing in Data Bases, Computer Journal, Vol. 15, No. 5, pp. 95～98 (1972).
9) C. R. DeFiore and P. B. Berra: A Data Management System Utilizing an Associative Memory, AFIPS Conf. Proc., Vol. 42, pp. 181～185 (1973).
10) C. R. DeFiore and P. B. Berra: A Quantitative Analysis of the Utilizations of Associative Memories in Data Management, IEEE Trans. on E. C., Vol. C-23, No. 2, pp. 121～132 (1974).
11) C.R. DeFiore et al.: Associative Techniques in the Solution of Data Management Problems, Proc. 1971 ACM National Conf. pp. 28～36 (1971).
12) M. Demartinics et al.: A Self-Managing Secondary Memory System, Third Annual Symposium on Computer Architecture, pp. 186～194 (1976).
13) M. Edelberg and L. R. Schissler: Intelligent Memory, AFIPS Conf. Proc., Vol. 45, pp. 393～400 (1976).
14) L. D. Healy et al.: The Architecture of a Context-Addressed Segment Sequential Storage, Proc. FJCC, Vol. 41, pp. 691～701 (1972).
15) IMSAI 108 Intelligent Disk System User's Mannual, IMS Associates, Inc. (1975).
16) C. S. Lin et al.: The Design of a Rotating Associative Memory for Relational Data Base Management Applications, ACM TODS, Vol. 1, No. 1, pp. 53～65 (1976).
17) R. R. Linde et al.: Associative Processor Applications to Real-Time Data Management, AFIPS Conf. Proc., Vol. 42, pp. 187～195 (1973).
18) G. J. Lipovski and S. Y. W. Su: On Non-Numeric Architecture, SIGARCH Vol. 4, No. 1, pp. 14～29 (1975).
19) H.H. Love: An Efficient Associative Processor Using Bulk Storage, Proc. 1973 Sagamore Computer Conf., on Parallel Processing (1973).
20) T. Marill and D. Stern: The Data computer ——A Network Data Utility, AFIPS Conf. Proc., Vol. 44, pp. 389～395 (1975).
21) D. R. McGregor et al.: High Performance Hardware for Database, Systems for Large Data Bases, North-Holland Publishing Co. (1976).
22) J. Minkey: An Overview of Associative or Content Addressable Memory Systems and a KWIC Index to the Literature: 1956～1970, Computing Reviews, Vol. 12, No. 10 (1971).
23) N. Minsky: Rotating Storage Devices as Partially Associative Memory, AFIPS Conf. Proc., Vol. 41, pp. 587～595 (1972).
24) R. Moulder: An Implementation of a Data Management System on a Associative Processor, AFIPS Conf. Proc. Vol. 42, pp. 171～176 (1973).
25) E. A. Ozkarahan et al.: RAP An Associative Processor for Data Base Management, AFIPS Conf. Proc., Vol. 44, pp. 379～387 (1975).
26) E. A. Ozkarahan et al.: Performance Evaluation of a Relational Associative Processor, Technical Report CSRG-65, Computer Systems Research Group, University of Toronto, (Jan. 1976).
27) B. Parhami: A Highly Parallel Computer System for Information Retrieval, Proc., FJCC, Vol. 41, pp. 681～690 (1972).
28) R. Peebles and E. Manning: A Computer Architecture for Large (Distributed) Data Bases, Proc., International Conf. on Very Large Data Bases, pp. 364～1375 (1975).
29) J. A. Rudolph: A Production Implementation of an Associative Array Processor-STARAN, Proc., FJCC, Vol. 41 (1972).
30) S. A. Schuster et al.: A Virtual Memory System for a Relational Associative Processor, AFIPS Conf. Proc., Vol. 45, pp. 855～862 (1976).
31) D. L. Slontnick: Logic per Track Devices, Advances in Computers, Academic Press (1970).
32) S. Y. W. Su and G. J. Lipovski: CASSM: A Cellular System for Very Large Data Bases, Proc. International Conference on Very Large Data Bases, pp. 456～472 (1925).
33) K. J. Thurber and L. D. Wald: Associative and Parallel Processors, Computing Surveys, Vol. 7, No. 4, pp. 215～255 (1975).
34) 有澤 博, 土肥康孝: ハードウェア・ソートによるデータベース基本演算の高速化について, 電子通信学会電子計算機研究会資料 EC 76～79.

35) 石井 治: 電子ディスク, 情報処理, Vol. 17, No. 12, pp. 1160～1160 (1976).
36) 植村俊亮: データベース・マシンのアーキテクチャ, 昭和 51 年電気四学会連合大会 (1976).
37) 植村俊亮他: 磁気バブルによるデータベーマスシンの構想, 電子通信学会電子計算機研究会資料 EC 76～78.
38) M. Maekawa, Loose or tight——a dilemma in the design of multi-mini-processor systems, 第 17 回情報処理学会全国大会, pp. 67～68 (1976).
39) 田畑隆司他: 並列処理にもとづく Relational Database Machine のアーキテクチャ——HARPS の応用——, 第 17 回情報処理学会全国大会, pp. 47～48 (1976).
40) 関野陽: 非数値処理アーキテクチャ会議に出席して——DBMS のハードウェア・サポートの研究, 情報処理学会データベース研究会資料. 29 (1976).
41) 関野陽, 植村俊亮: データベース・マシン, 情報処理, Vol. 17, No. 10, pp. 940～946 (1976).
42) 田中哲男他: ポリプロセッサ・オペレーティングシステム: EPOS の構造, 第 17 回情報処理学会全国大会, pp. 329～330 (1976).
43) 土井喜一, 関口 弘: 分散型データベース, 情報処理, Vol. 17, No. 10, pp. 934～939 (1976).

(昭和 52 年 1 月 8 日受付)
(昭和 52 年 2 月 16 日再受付)