データベース・システム 63－4
（１９８８．１．１８）

# 並列型データベースマシンにおける結合処理

*上林 弥彦　*中村 千秋　**石田 真美
*九州大学･工学部
**富士ゼロックス

マルチプロセッサシステムを用いたデータベースマシンの研究における中心課題は関係データベースにおける結合や、整列などの操作を実現する処理時間の少ない方法を求めることである。これらの操作についてはいろいろな手法があるが、本稿では一般に知られているマージソート法の均質型マルチプロセッサ向きのアルゴリズムを示す。結合操作を専用のハードウェアにより行う場合、データの性質が利用されず、無駄な処理を行ってしまうようなことも多い。そこで、本稿では、すでにソートされたデータについては、その後の更新部分を分けることにより効率を向上させる結合アルゴリズムを提案する。さらに比較のためSABREシステム上でパラメータを用いた解析式によって解析する。

# Join Operations for Homogeneous Multi－Processor Database Machines

Yahiko KAMBAYASHI*, Chiaki NAKAMURA* and Mami ISHIDA**

* Dept. of Computer Science and Comm. Eng., Kyushu Univ., Fukuoka, 812 Japan
** System Technology Research Lab., Fuji XEROX, Shinjuku, Tokyo, 160 Japan

Recently research on Data Base Machines utilizing multi-processor systems is getting popular. One of the major objectives is to reduce processing cost to perform join and sort operations in relational databases. There are various methods to handle these operations. This paper discusses marge-sort based algorithm suitable for homogeneous multi-processor systems. Conventional methods utilizing join hardware, usually do not use some properties of data. Thus, we also propose a join algorithm utilizing the result of the previous sort. Furthermore, we apply our approach to the SABRE system and discuss performance improvement on that system.

## 1．まえがき

本稿は、均質型のマルチプロセッサデータベースマシンに適した結合処理の方式について検討し、その結果をSABREシステムに適用して、どの程度の改良が可能であるかを示したものである。

データベースシステムの各種の処理の中で最も処理時間のかかるのが結合処理である。結合処理の実現方式は種々のものが知られているが、System Rの開発時の研究によれば、ソートマージ法と入れ子ループ法とが特に優れているという結果が得られている[1]。

すなわち、結合するべき２つの関係の大きさの差がかなりある場合は入れ子ループ法が優れており、それ以外の場合はソートマージ法が優れている。複数プロセッサを用いたソートマージ法としてはToddのもの[4]や田中のもの[3]が知られている。これらの方法は木構造のソートマージの各段に１つのプロセッサを対応させてパイプライン処理するものであるが、システムのメモリ使用の量に大きな偏りがあるという問題がある。一般的な均質型のマルチプロセッサシステムを用いた場合、ソートマージ法よりも、入れ子ループ法の方が並列処理に適しているため、単一プロセッサの場合よりも入れ子ループの適している場合が多くなっている[2]。しかし、基本的な処理量の総和はソートマージ法の方が少なくなる場合が多いため、このような場合のソートマージ法を改良することが重要といえる。本稿では以下に述べる２つの方法でソートマージ法の改良を行った。

１つの方法は、ソートマージ法の並列性の向上である。ソートマージ法では、特に２つのソートされた関係を結合する場合にほとんど並列性が生かせなくなるため、各関係のソートと結合の２つのステップを同時に実行することや、b-列マージのbの値を動的に変更して並列性を向上させることを試みる。

入れ子ループ法は本質的にデータが並んでいることを利用しないものであるため、データが並んでいるときにはソートマージ法の方が有利となる。関係の結合については良く用いられる結合があり、そのようなものでは前に結合してからの変更部分のみに注目すれば結合の効率を上げられると考えられる。このような特性の利用にはソートマージ法の方が適合している。また、この場合にはデータの分布が変更部分もほぼ同じであると仮定すれば並列性の高い分散ソート法も用いることができる。

本稿では、これらの方法について検討し、次にSABREシステムに適用して、従来知られている方式との効率の比較をパラメータによる解析式によって行う。

## 2．基本的事項

関係データベースは関係の集合で構成され、各関係は一枚の表に見立てることができる（図１）。この表の縦の列を各々を属性と呼び、横の行を組と呼ぶ。関係Ｒを構成する構成する属性集合を関係スキーマと呼び、Ｒと表記する。

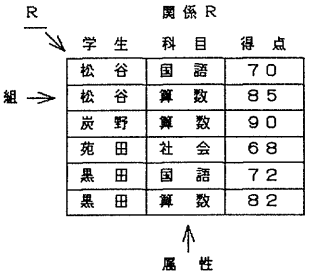



図１　関係Ｒ

関係データベースにおける関係演算には次のようなものがある。

射影：　関係の表の列のうち、必要な列のみを取り出す操作をいう。

選択：　表の中から、ある条件を満たす組だけを取り出す操作をいう。

整列：　関係のある属性に注目し、その値で順序よく並べ換えてしまう操作をいう。

結合：　二つの関係に共通な属性によって、その２つの関係を１つの関係につなぎ合わせる操作をいう。

結合操作は整列と共に処理時間を最も費やす部分である。そのためにデータベースマシンの性能評価に使われる。本稿では、この操作の効率化を図るための手法を提案する。以上の操作を図２に示す。

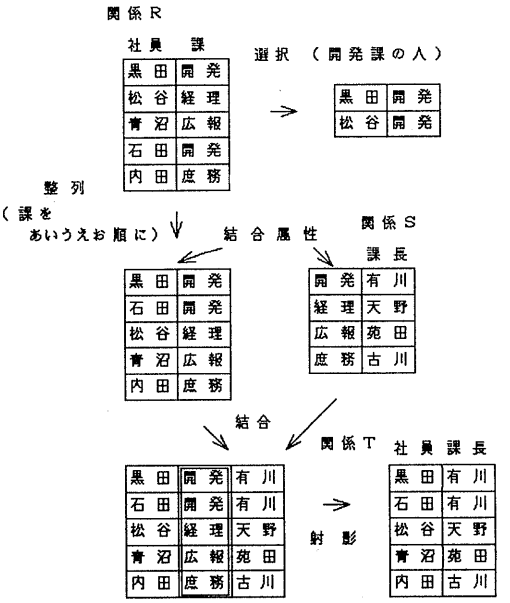



図２　結合演算

結合処理の手法としては、種々のものがあるが、入れ子ループ法、ソートマージ法などがよく知られている。単一プロセッサ型のシステムにおいてはソートマージ法が効率よくできると知られている。しかし、マルチプロセッサ型のシステム上では、並列性を考えるといちがいには言えない。結合処理の手法として上述の入れ子ループ法、ソートマージ法についての説明を行う。

入れ子ループ法

この手法は、２つの関係の突合せを繰り返すことによって行われる。まず、結合を施される２つの関係のうち、大きい方の関係を外部関係と呼ぶ。また、小さい方の関係を内部関係と呼ぶ。

手順としては

1.内部関係から１つの組集合を取り出す。

2.外部関係の組集合を順次選び、内部関係の組集合との比較を行い、結合を行う。

3.できた結果を出力する。

4.以上の手順を内部関係の全ての組集合について行う。

以上の手順において各組集合が１つの要素からなる場合を図３に示す。

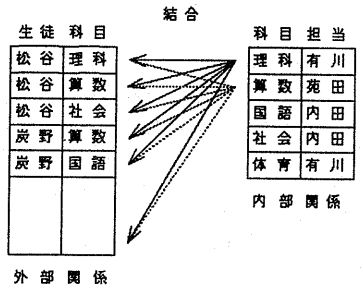



図３　入れ子ループ法

この手法では繰り返しの回数が内部関係と外部関係の組の数の積となる。このように効率が悪いために、片方の関係が小さい時は良いが、そうでない場合は単一プロセッサ型のシステムには向いていない。

ソートマージ法

この手法は、結合する２つの関係の各々を整列するステップと、前のステップで整列した２つの関係をマージするステップからなる。

・整列ステップ

まず片方の関係を整列させる。整列される前はデータの長さが１のｎ個の連の集まっている状態である。連というのはデータが並んだ状態の列のことである。これらの連をマージしていき、長さｎの１個の連にしてしまうわけである。このためにb-列マージを行う。手順としては

1.ｎ個の連をｂ個ずつに分ける。

2.ｂ個の連を比較し、マージする。この結果、長さｂのｎ／ｂ個の連ができる。

3.できた連を再びｂ個ずつに分け、マージする。

ｉ回目では長さ$b^{i-1}$の連ｂ個をマージして、長さ$b^{i}$の連がｎ／$b^{i}$個できる。

以上の処理を連の数が１個になるまで繰り返す。また、もう片方の関係についても同様の処理を行い整列してしまう。この処理を図４に示す。

・結合ステップ

このステップは整列ステップによって整列された２つの関係をマージする。この手順は、各々の関係を始めの方から比較していき（比較するべきデータを指示子で示す）、結合する。これを図５に示す。

この手法では関係の突合せの部分よりも、関係の整列の部分が利いている。この整列部分にはＯ（n･logn）ステップしかかかっていない。それで単一プロセッサ型のシステムにおいては入れ子ループ法より効率よく処理することができる。

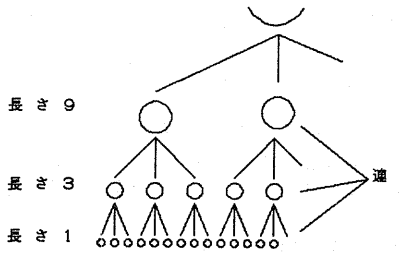



図４　整列ステップ

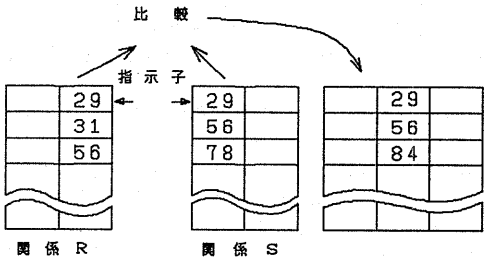



図５　マージステップ

## ３．並列ソートマージの効率化

前項で紹介した２つの手法を均質型の並列データベースマシンに用いた結果としては、ソートマージ法よりも、入れ子ループ法の方が適している場合が多いという結果が得られている[2]。均質型の並列データベースマシンに対するソートマージ法の問題点として次の３点が考えられる。

（１）整列ステップにおいて、整列を行うときｂ分木ができるが、２つの関係を別々に整列するために２つの木を作らねばならず、非常にコストがかかってしまう。

（２）整列ステップにおいて、処理が進んでいくと、ある時点からマージに必要なプロセッサ数が、実際のシステムのプロセッサ数よりも少なくなり、働いていないプロセッサが増えてくる。これによって並列性が下がってしまう。

（３）マージステップにおいて、プロセッサを１つしか使用しないために、やはり並列性の低下を招いてしまう。

解決法

問題点（３）は、２つの関係を別々に整列することに起因する。これを組がどちらの関係に属するか見分けられるようにして、２つとも混ぜて整列することによって解決できる。こうすることによって、マージステップが必要でなくなる。この方法は等結合ばかりでなく、$\theta$結合にも使用できる。また、図６のように２つの木を作って処理する場合はプロセッサが遊ぶ部分はａ，ｂ，ｃの部分の和になるが、上述の手法によってｄの部分だけになる。

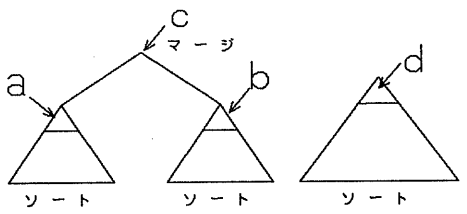



図6　マージの状態

さらに、dの部分を小さくするために、一度にマージする連の数を変更することによって並列性をあげることが可能である。

本稿では、この問題を解決する手法として、マージソート結合法を提案する。この手法の手順としては

1.２つの関係の組を混合する。但し、組にはフラグか何かでどちらの関係に属するか区別できるようにしておく。

2.ソートマージ法の整列ステップと同様の処理を行い、組の集まりを整列してしまう。

以上の処理によって２つの関係を結合できたことになる。この手法の利点を次にあげる。

・この手法は１つのｂ分木しか作らないのでプロセッサの遊ぶ部分は図５のｄの部分１ヶ所であるから整列マージ法に比べ並列性が向上している。

・この手法は等結合だけでなく、θ結合についても有用である。

・この処理を施された関係は常に整列しているために、次に結合や、整列などの処理要求がきた場合に便利である。

一方、ソートマージに基づく並列ソート法としてはTodd[4]、および田中[3]のものがよく知られている。これらの方法はパイプライン性を利用しているが、各プロセッサの使うメモリ量に大きな開きがあり、均質型並列データベースマシンでは使えない。均質型のものでは、各プロセッサを整列以外の用途にも使えることや、障害対策上優れているので、本稿では均質型のもののみを扱う。

## ４．変更されない部分の利用

これまでにあげた手法は、関係がどんな状態にあっても、それについては無視していた。しかし、実際はデータの更新を含め、次のような状態が考えられる。

ａ：結合属性で２つの関係とも、一度、整列、結合等の操作を受けている場合。

ｂ：結合する関係の片方のみが一度、整列、結合等の操作を受けている場合。

ｃ：結合する両方の関係とも一度も整列、結合などの処理を受けていない場合。

上述の各場合に対する解決法として次のものを提案する。

ａ：　更新部分以外はすでに整列した状態になっている。この性質を利用し、結合、整列の処理時間を短縮することができる。それは２つの関係のそれぞれの更新部分を取り出し、更新部分のみに整列操作を行う。これにより関係の一部しか整列を行わないために処理時間の短縮を図ることができる。その後、各々の関係からできる未更新部分と更新部分の４つの部分をマージすることで結合結果が得られる。また、１つの関係の未更新部分と更新部分をマージすることで整列した結果を得ることもできる。

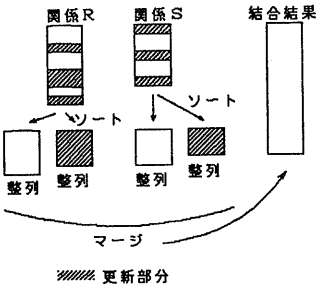



図7　更新率を考慮したソートマージ法

ｂ：　ａの場合と同じく、過去に整列、結合などの処理を受けた方の関係は更新部分を分離し、その部分にのみ整列操作を行う。もう片方の整列していない関係についてはマージソートなどの手法を用いて整列を行う。この結果できた１つの関係からの未更新部分と更新部分ともう１つの関係をマージすることによって結合結果が得られる。

また、関係が非常に大きい場合の等結合においては、整列している関係から結合する際に該当しないデータがわかるために、結合しながら結合するもう片方の関係の組を減らすことができる。例えば、整列している方の関係の結合属性のデータが1,8,9,15,・・・・と並んでいたとする。この時、この関係に[2-7]のデータがないというデータを入れておくと、結合中にもう片方の関係から2,3などのデータを含む組を除外することができる。但し、この方法では、生じた結果をθ結合で使用することはできない。

ｃ：　①で述べた方法を用いることによって結合することができる。

ａ，ｂの場合で、１つの関係が２つ以上の属性で整列させなければならない場合がある。データベースシステムでは副次索引が用いられるが、データベースマシンでは要求に応じて整列のやり直しを行う。この場合、別の属性による整列情報が失われることになるため、ここでは別の属性によって整列し直した関係は別に記憶しておくものとする。

更新が行われた場合にも両方に更新データを加えることで検索に対処する。

ａ，ｂの場合を手法を並列データベースマシン向けにするためにマージの段階に分散ソート法も併用する。すなわち、整列した関係を結合するために区間分けによるマージを行う。これは次のような手順を取る。

3-1.整列している各関係を一度検索して、その統計的性質を求め、プロセッサ数だけの区間にデータの量が等しくなるように割当てを行う。

3-2.各プロセッサに割り当てられた区間のデータを、放送されたデータの中から取り込む。そして、その後マージソートを行う（図９）。

3-1の手順は、整列されていない関係ばかりである場合に

は、一部をサンプル的に整列させ、全体の傾向を知ることもできる。このようなマージの手法を用いることによって並列性を上げることができる。

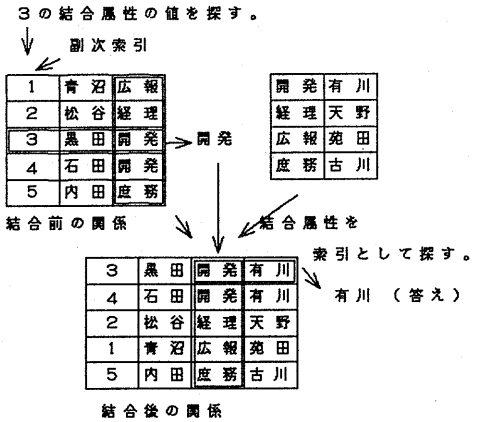



図８　結合属性による検索

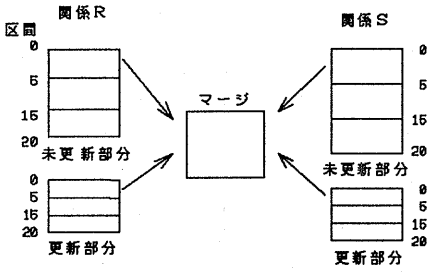



図９　区間分けによるソートマージ

## ５．均質型並列データベースマシンへの適用

システム

今まで述べてきた手法の比較のために文献[2]のシステム上で考察する。これはマルチプロセッサ型のシステムで、構成は図１０のようになっている。

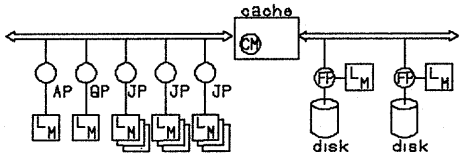



図１０　SABRE システム

このシステムの結合処理に関係する部分は次の部分である。

・FP（フィルタプロセッサ）
　ディスクから来るデータの選択を行う。
・キャッシュ
　ディスクから来るデータをページ単位で蓄える。
・JP（結合プロセッサ）
　データの結合処理を行う部分で、整列操作などもここで行う。このプロセッサは数ページのメモリからなるローカルメモリ(LM)を持つ。

また、このシステムの特徴として、次の２点があげられる。

・キャッシュから各ＪＰに対して放送機能を持つ。
・各ＪＰはキャッシュに対し、並列にデータをアクセスすることができる。

システムパラメータ

本稿では処理コストをその手法の処理時間で表す。このために次のようなパラメータを用いる[2]。

２つの関係RとSを結合する時に
m,n:関係R,Sの各々のページ数。
t:１ページのデータ数
b+1:JPの持つLMのメモリページ数
P:JPの数
JS:結合選択因子 (R join S)/(m•n)で定義される。

また、実行時間の評価に必要なパラメータは、I/O時間、プロセッサ間通信時間、CPU時間である。

・I/O時間

I/O操作時間として次の二つが考えられる。

Tdc:ディスクからキャッシュへのページ転送時間
Tcp:キャッシュからJPのLMへの転送時間

JPの平均読み取り時間CRはキャッシュヒット率Fによって決まるから次式によってあらわされる。

$$CR=F\cdot Tcp+(1-F)(Tcp+Tdc) \quad (1)$$

同様に平均書き込み時間CWはキャッシュに空きページフレームが存在する確率F'によって決まるから

$$CW=F'\cdot Tcp+(1-F')(Tdc+Tcp) \quad (2)$$

・プロセッサ間通信時間

ページ要求$M_{req}$、応答時間$M_{akn}$をI/O時間に加えると

$$CR\leftarrow CR+M_{req}+M_{akn} \quad (3)$$

$$CW\leftarrow CW+M_{req}+M_{akn} \quad (4)$$

また、プロセッサ割当てなどの制御メッセージはページ要求、応答に対して数も少なく、サイズも小さいので無視する。

・CPU時間

CPU時間として次の二つを定義する。
e:２つの属性の比較などの簡単な操作に必要な時間
i:LM内のページ内で、データを移動する時間

本稿では結合時間を次の二つの場合について考える。

1.結合されるページが整列していない場合

入れ子ループで結合する時間:CJ0

$$CJ0=t^2\cdot e \quad (5)$$

2.結合されるページが整列しており、マージ型演算で結合する時間:CJ1

$$CJ1=2t\cdot e \quad (6)$$

また、t個のデータからなるページの内部整列に必要な時間はtlogt回の比較と移動操作が必要であるから、１ページの整列時間CSは

$$CS=(e+i)t\cdot logt \quad (7)$$

・qページのマージ時間:CMq

qページの整列したデータのマージにはq(q-1)t回の比較とqt回の移動操作が必要である。これにqページの読取り、書込み時間を加えると

$$CMq=q(q-1)t\cdot e+q\cdot t\cdot i+q(CR+CW) \quad (8)$$

文献[2]では入れ子ループ法、ソートマージ法の解析式を次のようにして導出している。

入れ子ループ法

JPのLM内には放送用に1ページ、出力用バッファに1ページ用意しておく。関係Rを外部関係、関係Sを内部関係とする。つまりm>nである。各P個のJPは外部関係Rをそれぞれ(b-1)ページずつ並行に読み取る。外部関係Rを読み込んでしまうにはm/(P(b-1))回の繰り返しが必要である。

次に1回のパスにかかる処理時間を考える。まず、放送機能を使った内部関係Sの1ページをJPが読み取るにはCRだけかかる。この読み込んだ1ページと外部関係の(b-1)ページを入れ子ループ法で結合するために消費される時間は

$$CJO\cdot(b-1)$$

この結果生じるデータ数は(b-1)JSページである。この結果を出力バッファに移動する時間は

$$(b-1)JS\cdot t\cdot i$$

出力バッファがいっぱいになるとキャッシュへ送らなければならないが、この時間は

$$(b-1)JS\cdot CW$$

これを内部関係Sを全て放送で流してしまうまで行われる。これらの処理時間を足し合わせると1回のパスにかかる処理時間を求めることができる。

1パスにかかる時間

$$=(b-1)CR+n(CR+CJO(b-1)+(b-1)JS\cdot t\cdot i+(b-1)JS\cdot CW \quad (9)$$

以上の処理をm/((b-1)・P)回行うから、入れ子ループ法による結合処理時間TIME(NL)は

$$time(NL)=m/p\cdot CR+m/((b-1)\cdot P)\cdot n\{CR+CJO(b-1)+(b-1)JS\cdot t\cdot i+(b-1)JS\cdot CW\} \quad (10)$$

ソートマージ法

・ソートステップ

まず、関係Rを整列することを考える。LM内には1ページの出力用バッファを用意しておく。また、関係の大きさmは

$$modulo(m,P)=0 \quad (11)$$

とする。これは各JPが等しく仕事を分担することを表す。

まず、各JPがLM内にデータを取り込み、内部整列を行う時間は、1つのJPが受け持つデータはm/Pであるから

$$m/P\cdot CS$$

最初のパスでは、1ステップで1ページのサイズのb個の連をマージしてbページの長さの1個の連にする。k番目のステップでは$b^{k-1}$ページの長さのb個の連をマージし、$b^k$の長さの1個の連を作る。できた連の数をMとすると、M>P・bであれば、各JPは1ステップでb個の連を作るから、全ての連を読み取るためにM/(P・b)ステップ必要である。各パスで必要な時間は

$$m/(P\cdot b)\cdot CMb$$

この処理を進めていくとM=P・bになるパスがくる。これをオプティマルステージと呼ぶ。これより前のパスでは各JPは常時稼働することができる。オプティマルステージまでのパスの回数は$b^k=m/P$であるからk=$\log_b(m/P)$である。オプティマルステージまでに必要な処理時間は

$$m/P\cdot CS+m/(P\cdot b)\cdot CMb\cdot\log_b(m/P)$$

オプティマルステージはP個の連を生じる。P=bである場合、これらの連をマージするには1個のJPしか必要でない。P>bの場合、JPをb分木に配置し、マージを行う。これ以降のパスをポストオプティマルステージと呼ぶ。m/Pページからなる P個の連をマージするのに必要なパスは$\log_b P$である。ポストオプティマルステージでは必要なJPは1/bずつになり、1個の受け持つ仕事はb倍になっていく。最初のポストオプティマルステージではP/b個のJPが長さm/Pページのb個の連をマージするからこのパスで必要な時間$PO_1$は

$$PO_1=CMb\cdot m/P \quad (12)$$

i番目のパスにおいては長さ$m/P\cdot b^{i-1}$ページのb個の連をP/$b^i$個のJPでマージを行う。i番目のパスで必要な時間は

$$PO_i=CMb\cdot n/P\cdot b^{i-1} \quad (13)$$

ポストオプティマルパス全体で必要な処理時間POは

$$PO=CMb\cdot m/P+CMb\cdot m/P\cdot b+\cdots\cdots\cdots+CMb\cdot m/P\cdot b^{(\log_b P-1)}$$
$$=CMb\cdot m/P\cdot(1+b+b^2+\cdots\cdots\cdots+b^{(\log_b P-1)}) \quad (14)$$

ここで

$$1+b+b^2+\cdots\cdots\cdots+b^k=(1-b^k)/(1-b) \quad (15)$$

であるから

$$=CMb\cdot m/P\cdot(1-b^{\log_b P})/(1-b)$$
$$=CMb\cdot m/P\cdot(1-P)/(1-b) \quad (16)$$

mの大きさの関係をマージソートする時間をsort(m)とすると、関係Rを整列するのに必要な時間sort(m)は

$$sort(m)=m/P\cdot CS+m/(P\cdot b)\cdot CMb\cdot\log(m/P)+CMb\cdot m/P\cdot(1-P)/(1-b) \quad (17)$$

同様に、関係Sを整列するのに必要な時間sort(n)は

$$sort(n)=n/P\cdot CS+n/(P\cdot b)\cdot CMb\cdot\log(n/P)+CMb\cdot n/P\cdot(1-P)/(1-b) \quad (18)$$

・マージステップ

次に2つの整列された関係をマージする。単1プロセッサマージ型の処理は、2つの関係の読取り、結合、結果の書込みからなる。この処理時間merge(m,n)は

$$merge(m,n)=(m+n)CR+max(m,n)CJ1+m\cdot n\cdot JS\cdot CW \quad (19)$$

ソートマージ型結合における処理時間TIME(SM)は

$$time(SM)=sort(m)+sort(n)+merge(m,n)$$
$$=m/P\cdot CS+m/(P\cdot b)\cdot CMb\cdot\log(m/P)+CMb\cdot m/P\cdot(1-P)/(1-b)$$
$$+n/P\cdot CS+n/(P\cdot b)\cdot CMb\cdot\log(n/P)+CMb\cdot n/P\cdot(1-P)/(1-b)$$
$$+(m+n)CR+max(m,n)CJ1+m\cdot n\cdot JS\cdot CW \quad (20)$$

本稿でも文献[2]のパラメータを使用し、以下のような解析を行った。

マージソート結合法

文献[2]の手法の整列ステップのみをm+nの組に対して行うので

$$time(MS)=sort(m+n)=(m+n)/P\cdot CS+(m+n)/(P\cdot b)\cdot CMb\cdot\log((m+n)/P)+CMb\cdot(m+n)/P\cdot(1-P)/(1-b) \quad (21)$$

・ページサイズ最適化マージソート結合処理

3－①の問題の解決法でbの値を変更することによって並列性を向上できると述べた。しかし、bの値をむやみに小さくしてもかえってプロセッサ間通信時間の増加のために効率を落としてしまいかねない。bの値の変更によって影響を受けるのは、ポストオプティマルステージの部分で

ある。この処理時間を最小にするには、以下に上げる式を最小にするbを求めることである。

$$CMb/(b-1)=\{b(b-1)t\cdot e+b\cdot t\cdot i+b(CR+CW)\}/(b-1) \quad (22)$$

更新率によるソートマージ法

2つの関係R,Sの更新率をr,s(0<r<1,0<s<1)とする。すると関係Rの更新部分はr・m、関係Sの更新部分はs・nとなる。この更新部分をマージソートするのに必要な時間は式(17)より

$$sort(r\cdot m)=r\cdot m/P\cdot CS+r\cdot m/(P\cdot b)\cdot CMb\cdot log(r\cdot m/P)+CMb\cdot r\cdot m/P\cdot(1-P)/(1-b) \quad (23)$$

$$sort(s\cdot n)=s\cdot n/P\cdot CS+s\cdot n/(P\cdot b)\cdot CMb\cdot log(s\cdot n/P)+CMb\cdot s\cdot n/P\cdot(1-P)/(1-b) \quad (24)$$

これによって4つの連を生じる。これを区間分けし、各JPはマージする。各JPが受け持つデータは(m+n)/Pである。4つの連の比較に必要な時間は

$$4(4-1)max((1-r)m,(1-s)n)\cdot t\cdot e$$

マージした結果を出力バッファに移動する時間は

$$(m+n)\cdot t\cdot i$$

以上の結果よりマージする時間d_mergeは

$$d\_merge=1/P\cdot\{4(4-1)max((1-r)m,(1-s)n)\cdot t\cdot e+(m+n)t\cdot i+(m+n)(CR+CW)\} \quad (26)$$

よって全体の処理時間は

$$time(DMS)=sort(r\cdot m)+sort(s\cdot n)+d\_merge$$
$$=r\cdot m/P\cdot CS+r\cdot m/(P\cdot b)\cdot CMb\cdot log(r\cdot m/P)+CMb\cdot r\cdot m/P\cdot(1-P)/(1-b)$$
$$+s\cdot n/P\cdot CS+s\cdot n/(P\cdot b)\cdot CMb\cdot log(s\cdot n/P)+CMb\cdot s\cdot n/P\cdot(1-P)/(1-b)$$
$$+1/P\cdot\{4(4-1)max((1-r)m,(1-s)n)\cdot t\cdot e+(m+n)t\cdot i+(m+n)(CR+CW)\} \quad (27)$$

入れ子ループ法

$$time(NL)=m/p\cdot CR+m/((b-1)\cdot P)\cdot n\{CR+CJO(b-1)+(b-1)JS\cdot t\cdot i+(b-1)JS\cdot CW\}$$

ソートマージ法

$$time(SM)=sort(m)+sort(n)+merge(m,n)$$

マージソート結合法

$$time(MS)=sort(m+n)$$

更新率によるソートマージ法

$$time(DMS)=sort(r\cdot m)+sort(s\cdot n)+d\_merge$$

## 6．比較

比較のために各パラメータを次のように設定する[2]。

e=10μsec, i=250μsec, t=100
F=0.8, F'=0.3, Tdc=30msec
Tcp=4msec, Mreq=Makn=10msec
JS=0.01

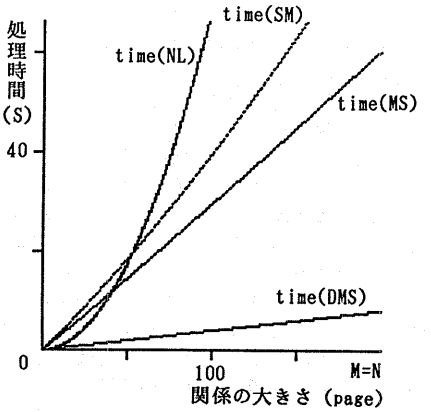
以下、グラフ内に出てくる記号は次の手法を表す。
time(NL) :入れ子ループ法
time(SM) :文献[2]のソートマージ法
time(MS) :本稿のマージソート法
time(DMS):本稿のデータの性質に着目した区間分けマージソート法

関係の大きさに対する処理時間

関係の大きさと処理時間の関係をグラフ1に示す。上記の手法の内、上から3つは、データがどんな状態にあっても処理を行える。グラフ1によりこれらの比較を行うと、次のようなことがわかる。

・本稿のマージソート法は文献[2]のソートマージ法と比べ、関係の大きさがいかなる場合にも処理コストを低く抑えることができる。

・入れ子ループ法は、関係が大きくなるにつれて、処理コストが他のものよりも大きくなってしまう。これは内部関係自体の大きさの増加と、外部関係の増加による内部関係の放送回数の増加によるものと思われる。




P=16, b=4, r=s=0.1

グラフ1 関係の大きさに対する処理時間

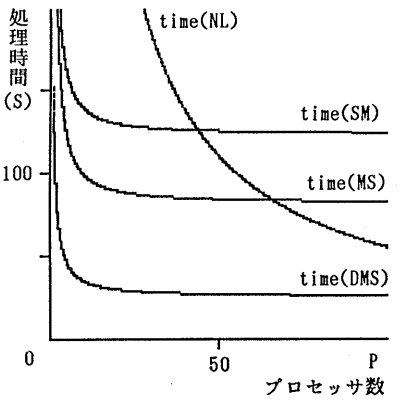
JPの数に対する処理時間

JPの数に対する処理時間の関係をグラフ2に示す。このグラフからわかることは

・JP数が多くなるにしたがい入れ子ループ法の処理コストが低くなっている。これはJPの増加によって一度に格納できる外部関係の容量が多くなるために放送の回数が減少するためだと考えられる。

・他の方法はある程度JP数が増えた時点で頭打ちを向かえてしまう。しかし、逆にみると、少ないJP数で処理コストを低く抑えることができる。これは、実際のシステムを作る場合、重要なことである。

この場合においても文献[2]のソートマージ法よりも、本稿の手法の方が優れていると言える。

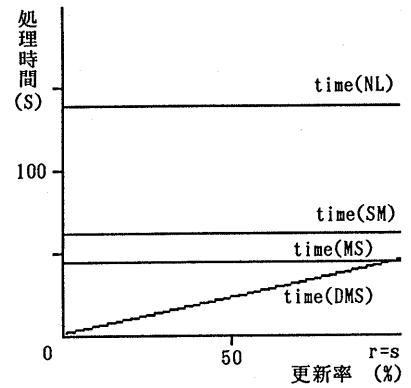



M=500, N=100, b=4, r=s=0.3

グラフ２　JPの数に対する処理時間

更新率に対する処理時間

更新率に対する処理時間の関係をグラフ３に示す。このグラフは当然、本稿のデータの性質に着目したマージソート法にのみ関係する。更新率が増加するにしたがい、処理コストも増加している。これは式(27)からもわかるように、更新率の増加による更新部分の整列処理コストの増加のためである。しかしながら、更新率が上がっても文献[2]のソートマージ法よりも効率の良いことがわかる。これは、区間分けによってマージを行うために、並列性が向上した結果のためであると考えられる。しかし、一般的な場合、ほとんどの部分が更新を受けるということはごくまれである。ほとんどの場合、更新を受ける部分は全体の数パーセントであり、データベースが大きくなればなるほどこの傾向は強いと思われる。このため、この様な手法は有用であると考えられる。しかし、１つの関係のある属性で結合した後、別の属性で結合するということが行われる。ここでは、二次記憶が充分大きいと仮定して、両方のソート結果を記憶しているものと仮定している。




P=16, b=4, M=200, N=100

グラフ３　更新率に対する処理時間

以上の結果より次のようなことが考察できる。構築しようとするデータベースが、非常に大きい場合にはソートマージ型の手法の方が適し、データベースマシン自体に対し、比較的小さい場合には入れ子ループ方が適する。

## ７．むすび

単一プロセッサの場合、入れ子ループ法に比べ、ソートマージ方が有利である。しかし、並列型システムの場合には入れ子ループ法の方が並列処理に向いているためにソートマージ法の適用範囲が減少するというのが文献[2]の主張であった。これに対し、本稿ではソートマージ法の改良を行い、適用範囲を広げた。また、これまでの手法とは異なったデータの並びという観点からデータの性質に基づいた手法を用いることで効率化を図り、データの中身によって処理方法を変えることの重要性を示した。

## 参考文献

[1] Seringer,P.G., Astrahan,M.M., Chamberlin,D.D.,Lorie,R.A. and Price,T.G. :Access Path Selection in a Relational Database Machine System :ACM SIGMOD, pp.23-34, 1979

[2] Valduriez,V., Gardarin,G. :Join and Semijoin Algorithms for a Multiprocessor Database Machine: ACM TODS, Mar. 1984

[3] Tanaka,Y., Nozaka,Y. and Masuyama,A. :PipelineSearching and Sorting Modules as Components of a Data Flow Database Computer,IFIP80,pp.427-432, 1980

[4] Todd,S.:Algorithms and Hardware for Merge SortUsing Multiple Processors, IBM J.R&D,22,5, pp.509-517, 1978