9L−2

パイプライン結合されたマイクロコンピュータによる関係データベースの結合演算

畢凱　大川善邦
大阪大学大学院　大阪大学工学部

1．はじめに

新しいデータベースシステムを提案する。データの格納場所として固定ディスク、コントローラとして高性能パーソナルコンピュータを用いる。データベースは関係データベースとし、そのなかで結合演算を検討する。処理時間を短縮するために、マイクロコンピュータをパイプライン状に接続した並列システムを考える。

データは固定ディスクにあるとし、そこからデータを取りだし、結果をディスクにしまうまでの時間を最短にすることを考える。コントローラはパイプラインの状態を監視し、状況に応じて処理の方法を変化させることによって、ディスクアクセスとパイプライン処理をバランスさせ、系の能力をフルに発揮させる。

2．並列システムの構造

マイクロコンピュータとバンクメモリを1次元状に接続した並列システムをマイクロコンピュータのパイプラインと定義する。ここで考えるパイプラインの基本形を図1に示した。パイプラインはコントローラの拡張バスに接続される。データはコントローラが所有する固定ディスク内に納められている。

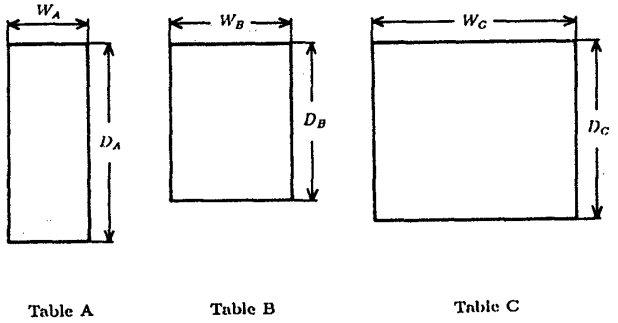

図2　結合演算に関係する表と記号

3．関係データベースにおける結合演算

ここでは、関係データベースの結合処理に必要となる記号について説明する。結合演算は2つのリレーション（これを以下で単に表と呼ぶ）から1つの表を導く処理である。図2に示すように、与えられた2表をA表、B表、結果として得られた表をC表と呼ぶ。各表を構成するタプルの数をD_A,D_B,D_Cとかく。各タプルの横幅をw_A,w_B,w_Cとかく。各表は等分割される。その各々をページと呼ぶ。n_A,n_B,n_Cは各表の総ページ数である。ここで結合率は$r=\frac{D_C}{D_A D_B}$によって定義する。

4．結合演算の手続き

コントローラはディスクより、A表のp個のページを取りだし、これを順々にパイプラインへ送る。パイプライン内のマイクロコンピュータは上流より流れてきたA表1ページを自己のローカルRAM内にに取り込む。既に取り込みが終わっているマイクロコンピュータは、そのページをそのまま下流へ転送する。転送終了時において、全てのマイクロコンピュータはローカルRAM内に、互いに異なるA表の1ページを持つ。

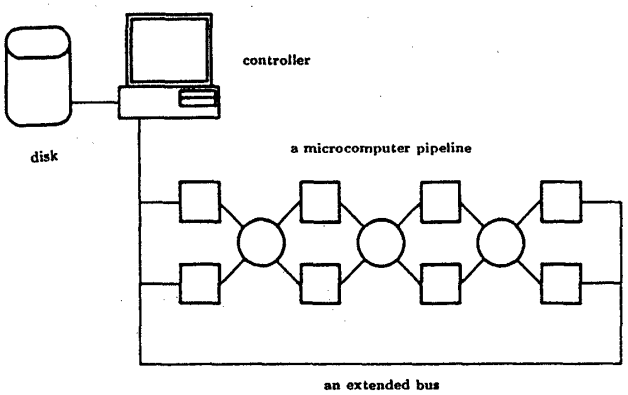

図1　データベース処理のパイプライン

Relational Database Join Operation in a Pipelined Microprocessors.
Bi Kai, Student, Graduated School, Osaka University
Yoshikuni Okawa, Faculty of Engineering, Osaka University

準備ができたパイプラインに対して、コントローラはB表を1ページづつ、最初から最後まで送り込む。各マイクロコンピュータは、自己のローカルRAM内のA表とバンクメモリを介して送られてきたB表に対して、結合演算を実行し、その結果をパイプラインの下流へ流す。結果はすべてコントローラに引き取られ、ディスクへ書き込まれる。

このとき、A表の残りページがゼロならば、そこですべての処理は終わったことになる。もし、残ページがゼロでなければ、再びA表の別のページを選び同様の手続きを繰り返す。これはA表すべてを尽くすまで行う。

この基本モデルによって処理を行ったときのクリティカルな部分は、マイクロコンピュータにおいてはA表1ページとB表1ページから結合演算によって、C表1ページを作るところ (t_{AB}) であり、コントローラにおいては、B表の1ページをディスクから読みだし (t_B) これをバンクメモリへ転送し、かつC表pページをバンクメモリからディスクへ書き込むところである。全体の処理時間は次式となる。

$$T = \frac{n_A}{p}(pt_A + n_B max(t_{AB}, t_B + pt_C)) \quad (1)$$

5．パイプラインの適応制御

制御のアルゴリズムは以下の通りである。

disk-bound ⇒ decrease n_A, n_B

pipeline-bound and not jam ⇒decrease n_A, n_B

pipeline-bound and jam ⇒increase n_A, n_B

詳細の説明は省略する。

6．実験

実際に並列処理装置を製作し、実機を用いてデータを採取した。データベースとして、大学の図書館の図書データを用いた。最初に、アルゴリズムを固定した状態で結合演算を行い、全処理時間を計測した。その結果は図3となった。表の組み合わせは6通りとし、結合率は全データに対する平均値を用いた。

次に、スタートは各アルゴリズムによって出発するのだが、ここで提案した適応制御アルゴリズムを用いて処理の方法を変更した場合の結果を図4に示した。図から明らかなように、どのようなアルゴリズムによってスタートしたにしても、全処理時間は最適値に収束している。

7．おわりに

この処理系は、ディスクアクセスがボトルネックになるか、パイプライン処理がボトルネックになるかのいずれかである。もし、ディスクがネックならば、パイプラインへ送るページのサイズを大きくし、ディスクへのアクセス回数を減少させる。もし、パイプラインがネックならば、パイプラインの処理の負荷を減らすためにページサイズを減らすのだが、その際に、パイプラインの中でバッファメモリのオーバーフローが起こらないように注意する。これが、ここで提案した適応制御の基本的な考え方である。この処理は、ソフトウエア的な手法のみで実現可能であり、ハードウエアの変更を必要としない。これが、本アルゴリズムの特徴である。

図3

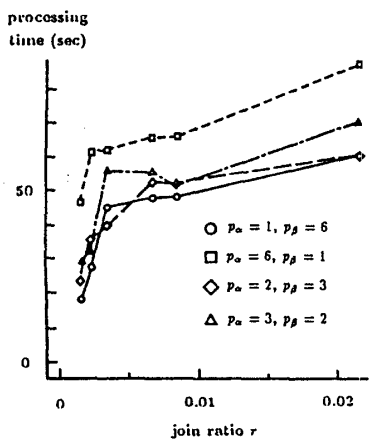

図4