4F-6

可変構造多重処理データベースマシンに於ける問合せ処理方式

鈴木重信・喜連川優・田中英彦・元岡達
(東京大学 工学部)

1. はじめに

一般に、並列処理可能な多重プロセッサシステムの上で関係データベースを処理する場合、単一プロセッサ上で処理する時とは異なるqueryの展開が必要となる。本マシンでは、ハッシングの手法によりバケットに分割されたリレーションを並列に処理できる。さらに複数個のオペレーションを並列に処理することも可能である。ここでは、その処理の様子を述べた後、あるqueryを最適なtreeに展開する手法について述べる。

2. Query処理の様子

a) 処理手順-------リレーションはディスクモジュールにページ毎に分割して格納されており、ステージングはページパラレルに行なわれる。ディスクモジュールにはフィルタプロセッサとハッシュユニットが置かれ、selectされたタプルが最初の処理に必要なアトリビュートに関してハッシュされて複数台のメモリモジュール(以下M.M.)にリングバスを介して送られる。この時同一バケット内タプルが各M.M.に平均して分散するような処理がなされる。M.M.はバケット毎にシリアルにタプルをプロセッシングモジュール(以下P.M.)に送出する。複数台のM.M.からの送出はパイプライン的に行なわれる。P.M.はあるバケット内のタプルのみを取り込み、その処理を行なう。(この処理は取り込みタプル数に比例した時間で終了する。) P.M.はディスクモジュールと同様にハッシュユニットを持ちM.M.に結果を返す際に次段のオペレーションに必要なアトリビュートに関してハッシュする。この結果の送出もステージング時と同様にリングバスを介して事前に割当てられた複数台のM.M.になされる。あるオペレーションの処理と次段のオペレーションのためのハッシュが重畳して行なわれる。以後同様に次段のオペレーションの処理となる。

b) メモリモジュールのマークビット管理-------M.M.は与えられたハッシュ値をマークビットにencodeして貯える。その際バケット毎にカウンタを持ち、バケット毎のタプル数を管理する。バッファループを持った改良型M/m方式の磁気バブルメモリの特性とマークビットによる連想読出し機能によりタプル送出の際あるバケットのみをほぼ連続して効率よく送出することができる。カウンタの管理によりあるバケット内タプルをすべて送出すると以後のバケットの送出にうつる。この送出は別のP.M.に対してなされる。

c) パイプライン効果-------2M台のP.M.(Mはリレーションを格納するM.M.台数)を駆動することによりパイプライン処理が可能となる。M.M.を2台とするとfig.1のようなパイプライン処理となる。これによりすべての関係代数演算はリレーションがバブル空間に入り得る限り、リレーションのサイズと関係なくO(m)の時間(mはM.M.の容量)で処理できる。

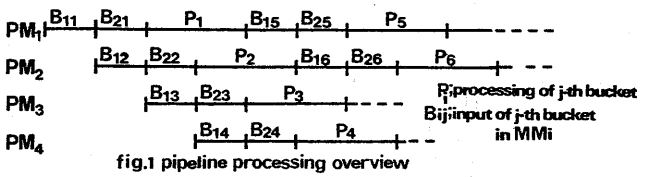

fig.1 pipeline processing overview

d) バケット調整--------バケット内タプル数をbとした時、各M.M.は同一のバケット内タプルをb/M個ずつ持つことが望ましく、この値がゆらぐことはパイプラインの擾乱、ソートモジュールのオーバフローにつながる。そのため適当なアルゴリズムを用いて各M.M.に平均して分散する様に処理する。又、bのゆらぎも同様の効果をもたらすため、一度細かいバケットを作り、それをいくつかまとめて大きなバケ

ットとする方式をとっている。

3. Query展開の指針

a) 無限資源の場合--------単一プロセッサの環境下ではINGRESの展開アルゴリズムが知られているがこれは逐次展開形である。本マシンの並列処理可能な環境下では資源が無限の場合、queryをバランスのとれた高さ最小のstrategy treeに展開するのが最も処理時間が少なくて済む。この時処理時間は$O(mh)$（hはtreeの高さ）である。

b) 有限資源の場合の最適展開のための要因--------本マシンの処理では処理時間はリレーションのサイズに関係ないため最適展開のための要因としては資源の有効利用のみが考えられる。各M.M.が単にアクティブなら必要な資源総量は初期リレーションと中間結果を格納するスペースである。そこで中間結果の総量最小が最適展開のための一つの要因となる。又M.M.にデータが格納されているにもかかわらず相手側リレーションが存在しないため処理待ちになる時間を最小にするのも一つの要因である。

c) 結果リレーションのサイズの予測--------中間結果最小の展開を実現するためには処理前に中間リレーションのサイズの予測が必要である。ここではアトリビュートの分布がある密度関数fをなすと見積れる場合、$d\left(1/d=\int f(x)^2dx\right)$という値を用いる。これによって結果リレーションのサイズをprojectionではd, selectionでは$|R|/d$, joinでは$|R_1||R_2|/d$というように予測する。（joinの場合アトリビュートが別々の分布をなす時$1/d=\int f_1(x)f_2(x)dx$でdを求める。）このdは一様分布ではdomainの範囲に相当する。

d) 逐次並列展開形--------M.M.のアクティブな時間を多くするためにはfig.2のような逐次並列展開形が有利である。即ち、M.M.に入り得る部分については並列に展開し、空きスペースができる毎に余ったリレーションを逐次的に結合していくが、その時中間結果と新リレーションで並列展開をはかる。リレーションの逐次結合の順序では処理結果が小さくなるような部分集合を選んでいき、並列展開ではバイナリ・オペレーションの結果が小さくなるよう2つの結合を選んでいく。

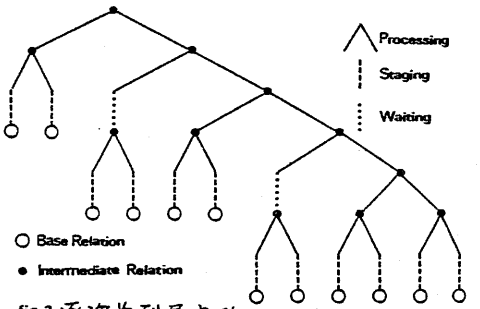

fig.2 逐次並列展開形
（各リレーションサイズは同じ。メモリ内に9リレーション入ると仮定。）

e) Query Networkからの展開--------実際のqueryでは任意のリレーションの結合が可能なわけではないので上のような方法が常に可能なわけではない。queryの静的な状態を表わしたnetworkの推移関係を考える。そこで今方（G_1）が与えられた資源に入りきるようなnetworkの2分割のうち、部分network（G_1, G_2）が両方とも連結になるようなものを考える。そのうちG_1の方の処理結果が最小となるものを最初に並列展開するためtreeのbottomとする。次段からは空きスペースに入るリレーションをG_1に含めて2分割を考え逐次的にtreeに取り込んでいく。並列展開では可能な組合せのうち中間結果が最小となるものを結合する。このような手法で最適展開形を求めていく。

4. おわりに

本マシンのアーキテクチャでのqueryの処理方式を述べたが、queryの最適展開についてはアーキテクチャに依存する要因が多い。しかし、有限資源での並列展開可能なアーキテクチャという一定のモデルに関しては逐次並列展開形という一つの方向があるということは本マシンに限ることではない。今後、有限資源での最適展開に関してより完全なアルゴリズムを開発していくつもりである。