4F-5

可変構造多重処理データベースマシンに於けるHashの適用

喜連川優　鈴木重信　田中英彦　元岡達
東京大学　工学部

1. はじめに

Logic per Trackから出発したRAPを中心とするCellular Logic Type DBMはSELECTION及びUpdateなどの演算に対しては従来のマシンに比べて大きな性能向上をもたらした。しかしO(N)をこえる処理負荷の重いJOIN, PROJECTIONに対してはその効果は薄くStream-Orientedな全数サーチ処理手法の適応限界が明らかにされたといえる。又DIRECTでもページレベルでやはりO(N×M)の処理負荷を要し、Hash Bit ArrayもJOIN候補の篩い落としには効果があるものの実際の結合操作には役立たない等、関係代数演算のO(N)化は充分に実現されているとはいえない。ここではHashとSortを用いたO(N)関係代数マシンについて報告する。

2. Hashの適用

Hashingはデータベースマシンに対する応用という立場では以下の2つの適用手法が考えられる。

a. CAFSに見られる篩い落とし[1]

b. クラスタリング

JOINの処理手法について考えるとaではバケット数を多くしてHashを施し両リレーション間でのシノニムを結合可能性のあるタプルとして以後の処理対象とする。この手法により両リレーションのサイズは減少し前処理効果は大きいものの実際の結合処理は対象外である。一方、bでは対象とするタプル数自体の減少を目的とするのではなく実際の結合処理における負荷を小さくしようとする手法である。即ち、JOINは最も単純に処理すればO(N×M)(M, N: 2つのRelationのCardinality)時間必要であるが両リレーションにHashを施し全体をSケのバケットに分割したとすると $N=\sum_{i=1}^{S} n_i$, $M=\sum_{i=1}^{S} m_i$ と表わされ所要時間は $T \propto \sum_{i=1}^{S} n_i \cdot m_i$ ですむ事になる。図1に示される如く大きな処理負荷の減少をもたらすことが明らかである。我々はこの2の手法により関係代数演算のO(N)化を試みた。

3. 並列化技法

Hashingにより、リレーションを互いに独立なバケットに分割する事が出来、処理負荷の大きな減少が実現されたが更に各バケットを並列に処理する事により、高速化が期待出来る。並列処理化に際しては以下の点を考慮する必要がある。

1) バンクパラレリズムの反映

複数のメモリバンクより並列にデータ出力可能であるならばm倍(m:バンク数)の高速化を実現する。

2) バケット内 O(m)化

2つのリレーションを各々互いに独立なケのバケットに分割する事によってバケットレベルでO(m)の処理負荷に減少出来たがバケット内での処理も、O(m)(m:バケットサイズ)化する。

3) プロセッサへのバケット分配技法

基本的には1つのバケットを1つのプロセッサに割当てるが、この際プロセッサに効率よく当該バケットのデータを収集する必要がある。又或程度以上の大きさをもつリレーションの処理を考えると全てのバケットを並列に処理する事は困難であり、バケットシリアルに、且、並列処理する必要がある。この場合データ流のもれを最小限にし、円滑なバケット切換を実現する必要がある。

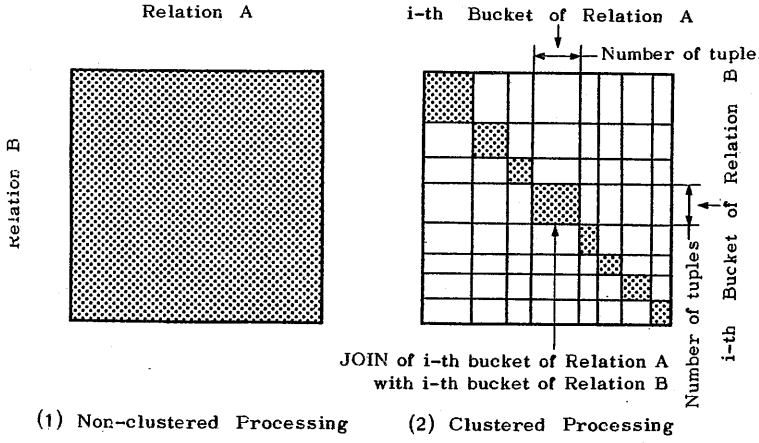


Fig. 1. Clustering Effects By Hashing In JOIN Operation

4) 分散の不均一性への対処

Hash関数はその性質上Hash後の分布は必ずしも均一にはならず、バケットサイズの偏りが激しい場合もあり、更にプロセッササイズに収まらず溢れをきたす事も考えられる。この分布不均一性は避け難い特性であり、効率よく処理する必要がある。

次に、これ等の問題点に対して、我々の採った手法について説明を加える。

1) メモリモジュール(或はディスクモジュール)がm台からなる時、m本のデータストリームに対し、2m台のプロセッサを駆動する事により、モジュール間のパイプラインを形成し基本的には N/m 時間で処理を完了出来る。

2) プロセッシングモジュール内では $\log N$ 台のサブプロセッサを用いたハードウェアソータにより $O(N)$ 時間のソートを実現している。この $O(N)$ ソータにより、バケット内での関係代数演算をほぼ $O(N)$ 時間で実現出来る。

3) メモリモジュールには石磁気バブルメモリを用い、マークビットRAMにより、Hash Bucket Idを管理する。M/m方式のバブルチップの改良により、当該バケットを構成するタプルをほぼ連続して出力することが可能となり、バケットシリアル処理を効率よくサポート出来る。又各バケットはモジュール間での平滑化を行ないプロセッサのデータ収集効率を高めている。

4) Hashを終えた後、各バケットをプロセッササイズに統合し、プロセッサの使用効率を高めるとともに、モジュール間パイプラインの撓みを押えている。バケット数が多いため、オーバフローは殆んど生じないが、起った場合には複数のプロセッシングモジュールを1つのチャネルを介し動的に結合する事で対処できる。

4. システムアーキテクチャ

本マシンのデータ操作部は図2に示される。

1) プロセッシングモジュール

本モジュールはソートモジュールと制御部からなり、前者は $O(N)$ ソートを実現するソーティングユニット及び条件式を評価するタプル操作部からなる。メモリモジュールからのデータ流をソートし、バケット入力完了後ソートされたデータに対し関係代数処理を施す。又結果タプルにHashを施す。

2) メモリモジュール

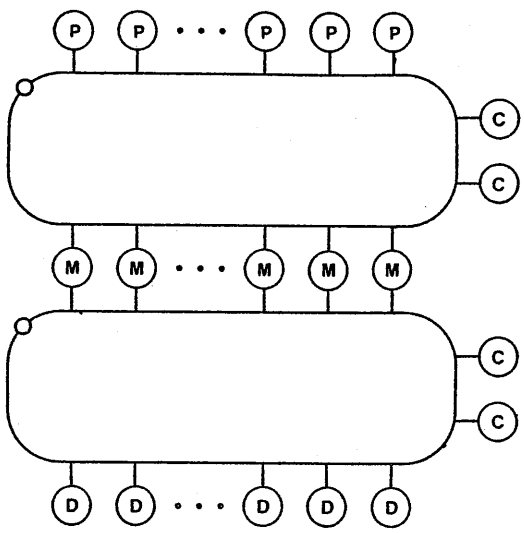
バッファループを有する改良型バブル[2]をメディアとし、バブル制御部、マークビット操作部、バケット管理部などからなる。バケットシリアルに効率よくデータをプロセッサへ出力する。

3) ディスクモジュール

本マシンは二次記憶系として、ディスクを用い、大容量データベースのサポートを可能にしている。本モジュールはディスク管理部、フィルタープロセッサ、ハッシングユニット等からなり、ディスク内データに対しRAP like のフィルターを通し、必要なタプルのみをメモリモジュールにステージングする。

4) コントロールモジュール

2つのリングに夫々存する本モジュールは上位システムから関係代数表現に変換されたQuery Treeを受けとり、リングバスチャネルを割り付けることにより、プロセッサとメモリを結合、駆動する。又、実行時のプロセッサ群制御並びにディスクからメモリに対するリレーションステージングを制御する。



P : Processing Module　C : Control Module
M : Memory Module　D : Disk Module

Fig.2. Global Architecture Of Data Manipulation Subsystem

5. おわりに

本稿ではHashとSortによる $O(N)$ 関係代数マシンの基本概念とシステム構成について述べた。実際の処理方式は次稿に譲る。我々は既に可変構造多重処理データベースの開発を行なって来たが[3]、現在リングバスインタフェースモジュールの製作によりマルチモジュール化を図るとともに、本方式の実装を進めている。

参考文献

(1) E.Babb TODS Vol4, No.1, 1979
(2) 高橋他 EC80-48, 1980 (3) 喜連川他 EC80-51, 1980