コネクションマシンCM-2上におけるハッシュジョイン処理

2P−4

松本 和彦　喜連川 優　高木 幹雄
東京大学 生産技術研究所

1 はじめに

コネクションマシンは、数万個というオーダーのプロセッサアレイとハイパーキューブ形状の通信ネットワークを持つ、単一命令ストリーム多重データ(SIMD)の並列アーキテクチャマシンである。現在Thinking Machines Corporationによる最新モデルはCM-2と呼ばれ、フルセットで65536個のプロセッサを備えている。

CM-2は、ワークステーションなどのフロントエンドにバスインタフェースを介して結合している。逐次的な処理はフロントエンド上で通常のプログラムコードとして実行され、並列処理の部分だけがバスインタフェースを通じてコネクションマシンに指令される。

CM-2の操作は、PARISというライブラリの関数を呼び出すことによって行なう。このライブラリがサポートしている重要な機能は、仮想プロセッサ(Virtual Processor, 以下VP)である。VPの数は物理的なプロセッサ数に関係なくとることができ、プログラマへの負担の軽減、プログラムの可搬性に大きく寄与する。VPの数を物理プロセッサの数で割ったものを、VP比と呼ぶ。

最近になって、PARISにsend-to-queueという新しい通信プリミティブが加わったので、それを用いてハッシュによる簡単なジョイン処理をインプリメントした。プロセッサ数8192、クロック6.7MHzのCM-2を使い、その評価を行なったので、結果を報告する。

2 ジョイン処理

関係データベースの基本演算の中でも、ジョイン処理は特に重要なものであり、同時に、負荷が大きい処理でもある。

これは、図1に示すように、2つのリレーションがあった場合に、同じキー属性どうしのタプルをつきあわせて、その直積を得る演算だと言うことができる。

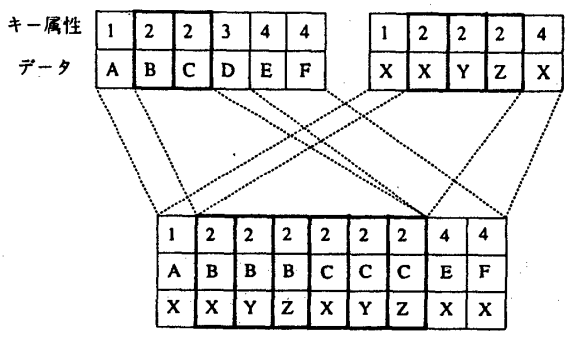

図1: ジョイン処理

ジョイン処理のアルゴリズムについては、さまざまなものが提案されているが、その代表的なものにハッシュによるものがある。

ハッシュによるジョインとは、キー属性を元にタプルをクラスタ分けする方法である。具体的には、キー属性の値をハッシュ関数にかけ、そのハッシュ値によってタプルをいくつかのクラスタに分ける。キーの比較はクラスタ内だけで行なえばよいので、全体としての比較回数は大きく減少する一方、同じキーを持つタプルは必ず同じクラスタに属するので、演算は誤りなく実行される。

3 send-to-queueプリミティブ

コネクションマシンでハッシュジョインを行なうためには、並列度を得るために、それぞれのクラスタを1つ1つのプロセッサに対応させることが効率的だと思われる。そのためには同じハッシュ値を持つタプルを同じプロセッサに送信することが必要であるが、1つのプロセッサに送信するメッセージの数が複数である場合には、メッセージの衝突の問題が生じる。今までのPARISライブラリに存在した通信プリミティブでは、このような衝突の扱いは、任意の一方とか加算・乗算などの算術処理といったように、いずれにせよ元のメッセージの一部が失われる方法だった。

新しいプリミティブsend-to-queueでは、いくつかの制約はあるものの、メッセージの失われない衝突処理が実現されている。送信先となるプロセッサでは、連続するメモリを配列として用意し、複数のメッセージが同じプロセッサに送信された場合は、着いた順にその配列の先頭から格納されていく。同時にプロセッサごとに用意されたカウンタの値がインクリメントされていく。通信が終了した時点で各プロセッサは自分のカウンタを参照して、自分にいくつのメッセージが送られてきたかを知ることができる。

このプリミティブにはいくつかの制約がある。まず第一に、当然のことながら配列のスロット数には限りがあるということである。その数はsend-to-queueの実行時に指定できるが、あふれたメッセージは失われる。ただし、カウンタは適切に更新される。また、第二に、スロットの大きさは32ビット固定ということである。これは、現行のsend-to-queueのインプリメントが未熟なためだと思われるが、実用にするためにはあまりに小さ過ぎる。

4 実装方式

ハッシュジョインの実際のインプリメントは、send-to-queueを使えば非常に自然にできる。まず、元のリレーションを持っているプロセッサは、タプルのキー属性から、各々のハッシュ値を計算する。ハッシュ値としてプロセッサのアドレスが生成するようにしておけば、タプルをハッシュ値を送信先アドレスとして送ることで、各々のプロセッサにクラスタ分けすることができる(図2参照)。

次のステップでは、それぞれのプロセッサは独立にネストループと呼ばれるアルゴリズムによってジョイン処理を行なう。このステップはクラスタ間の相互作用が無いので、完全な並列処理で行なうことができる(図3参照)。

---

[0]Hash join operation on the Connection Machine CM-2
K.Matsumoto, M.Kitsuregawa, M.Takagi
The Institute of Industorial Science, University of Tokyo

最後に、このままではプロセッサごとに保持するタプルの数が違うので、これを適切なプロセッサに送信することによって1プロセッサ1タプルの状態にする（図4参照）。

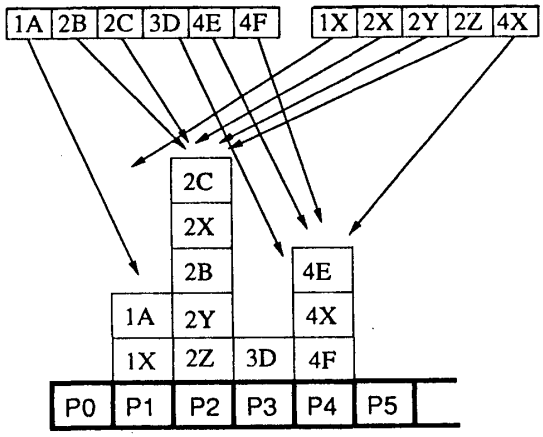

図2:

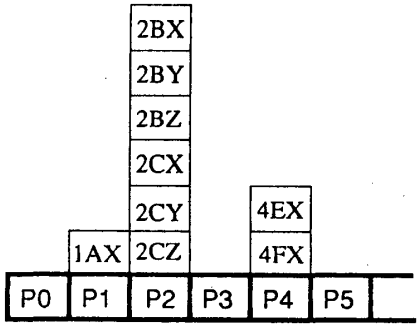

図3:

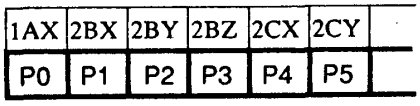

図4:

5 性能評価

実際にハッシュジョインを行なってみた結果を表1に示す。

はじめ、元となる二つのリレーションのそれぞれに「タプル数」の欄の数のタプルが保持されており、そのキー属性は0から'タプル数-1'のものが1つづつ、順序はランダムである。従って、生成される新しいリレーションでも、0から'タプル数-1'のキー属性を持ったタプルが1つづつ存在することになり、タプル数は元のリレーションに等しくなる（選択率100%のジョイン）。

測定対象としたのは、元となるリレーションが既にCM-2プロセッサにロードされている状態から、図4に示すようにジョイン後のタプルが各プロセッサに1つづつ保持されている状態になるまでに必要とされる時間である。

先に述べたがsend-to-queueでは32ビットのメッセージしか使えないため、タプルの長さは32ビット以内でなければならない。実際には作業用にそのうち2ビットを使っているので、キー属性が15ビット、データが15ビットという、非常に小さなタプルのジョインになっている。

使用したCM-2は、プロセッサ数8192個、クロックスピード6.7MHzのものである。

タプル数	処理時間 (msec)
512	55.4
1024	57.2
2048	59.2
4096	77.6
8192	98.6

表1: ハッシュによるジョイン処理時間

6 終りに

超並列 (Massively Parallel) マシンの特徴として、扱うデータ量が何倍になろうとも、基本的には定数時間で計算が行なわれることが挙げられる。当然、通信の時間はそうはいかないので、完全な定数時間とはならないが、非常にゆるやかな性能劣化となるのが普通である。今回の性能測定においてもその傾向は見てとることができ、タプルの数を10倍にしても処理時間は2倍にもならないことがわかる。

send-to-queueはまだできたばかりの（正確にはまだ実験段階の）プリミティブであるので、非常に制約の大きい面もあるが、これからの改良によって非常に有用なものになる可能性が大きい。今後はその改良を待ち、あるいはメッセージ送信を何回かに分けるなどして、より大規模な処理を行なうことが課題となるだろう。

参考文献

[1] W.Daniel Hillis, "The Connection Machine",The MIT Press,1985

[2] Thinking Machines Corporation, "Connection Machine Technical Summary",1989