5H−4

コネクションマシンCM2上における大規模関係データベース処理の性能評価

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

これまで、関係データベースシステムの性能を改善するための研究が、数多くなされてきた。その中には、アルゴリズムの改良、専用ハードウェアの開発などが含まれているが、汎用の並列アーキテクチャマシンを利用するというのは、よく行なわれる重要なアプローチの一つである。

アーキテクチャから見て、並列マシンは大きく二種類に分類することができる。ハイパーキューブマシン (iPCS、NCUBE) 、マルチステージ結合マシン (BNN butterfly) 、バス結合共有メモリマシン (Sequent、Aliant) などを含むMIMDマシンと、コネクションマシン、MPP、MasPar などを含むSIMDマシンである。これらのSIMDマシンは、小さなプロセッサを数多くという基本理念から、超並列マシン (Massively Parallel Machine) とも呼ばれる。

並列マシンと関係データベースシステムという関係から見ると、MIMDマシンについては、これまでにも多くの研究がなされている。しかし一方で、コネクションマシンなどの超並列マシンによる研究は、今のところほとんど報告されていない。

本論文では、我々が考案し実装したコネクションマシン上での大規模関係データベース処理について、処理時間の詳細を解析した結果を報告する。

2 コネクションマシン

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そのピーク演算性能において世界最高を誇っている。

コネクションマシンのプログラミングのためには、何種類かの環境が用意されているが、基本的なインターフェースとして、PARISと呼ばれるライブラリがC、Lispなどの言語用に用意されている。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。

一つのデータ集合に割り当てられるVPの集合のことをVPセットと呼び、各々のVPセットは任意数のVPを含むことができる。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。各々のVPセットは、各自のVP比を持つ。プログラムは複数のVPセットを扱うことで処理を進めることができる。

コネクションマシンのプロセッサアレイには直接CMIOバスと呼ばれる入出力バスが結合されており、大容量ディスク、カラーディスプレイなどの周辺機器をつなぐことができる。

[0]Performance Analysis of Large Relational Database Processing on the Connection Machine CM-2
K.Matsumoto, M.Kitsuregawa, M.Takagi
The Institute of Industorial Science, University of Tokyo

特にディスクはDataVaultと呼ばれ、最高8個までCMIOバスによってプロセッサアレイにつなぐことができ、並列に入出力を行なうことができる。

3 CM-2上のジョイン処理プログラム

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。

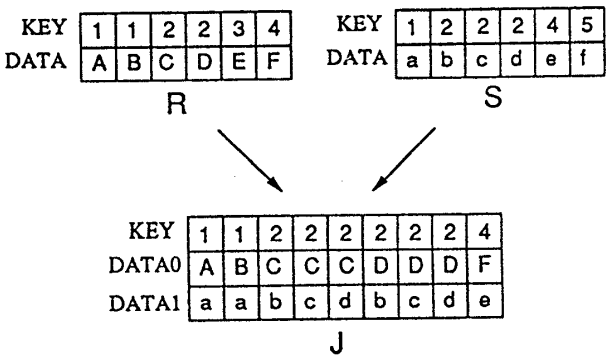

図1: ジョイン処理の例

本プログラムで用いたアルゴリズムは、大きく2つの部分に分かれる。スプリット処理の部分とジョイン処理の部分である。

一般に、データベースのサイズはマシンの実装メモリに比べてはるかに大きいので、そのままメモリ上でジョイン処理を行なうことはできない。それを解決する方法の一つとして、データベースをバケットと呼ばれる単位に分解するスプリットがある。ジョインはキー属性の同じタプルどうしをつき合わせれば良いので、キー属性によって分割を行なったそれぞれのバケットどうしでジョイン処理を行なえば、データベース全体でのジョイン処理を行なうことができる。

我々は、コネクションマシンCM-2上にスプリットジョインのプログラムをPARISライブラリを用いて実装した。CM-2は大規模二次記憶DataVaultを持っており、これを利用してスプリット処理を行なうことが可能である。アルゴリズムと実装の詳細については、[1]を参照されたい。

4 コネクションマシンのディスク入出力

コネクションマシンからDataVaultを使う方法は、逐次マシンから通常のディスクを用いる方法とほぼ同様である。違うのは、DataVault上のファイルに対して、コネクションマシンの全プロセッサが一斉に入出力を行なうということである。例えば64Kプロセッサのコネクションマシンだったら、1バイトwriteするという操作が、DataVault上に64Kバイトの情報を蓄えるということを意味する。

具体的にコネクションマシンのプログラムの中から

DataVaultを使うには、次のような手順を用いる。

1. CMFS-openプリミティブによって、DataVault上のファイルをオープンする。
2. CMFS-lseekによってファイルポインタを適切な位置へ移動させる。この時、ファイルポインタをnにセットするということは、(n×プロセッサ数) ビットのデータをスキップすることを意味する。
3. CMFS-read-fileまたはCMFS-write-fileによって、入出力を行なう。転送量はビット長で指定するが、それは1プロセッサあたりの転送量である。
4. CMFS-closeによってファイルをクローズする。

大量のデータをDataVaultから読み込んで逐次的に処理を行なう場合、全ての入出力に先だって全体の転送量を与えると、その後の入出力処理の一部をプロセッサの処理とオーバーラップさせることが可能である。これはストリーミング入出力と呼ばれ、我々のプログラムではスプリットされたバケットを読み込む時に用いている。具体的な手順は以下のようになる。

1. CMFS-openプリミティブによって、DataVault上のファイルをオープンする。
2. CMFS-fcntlによって、ストリーミング入出力を起動する。
3. CMFS-read-fileまたはCMFS-write-fileによって、全体の転送量を指定する。
4. CMFS-streaming-iostatによって、ストリーミング入出力の起動にエラーが無かったかどうか確認する。
5. CMFS-partial-read-fileまたはCMFS-partial-write-fileを繰り返し、プロセッサメモリへのデータ転送を行なう。このプリミティブとプリミティブの間に、プロセッサの処理をオーバーラップさせることができる。
6. CMFS-streaming-iostatによって、全ての転送が終了したかどうかを確認する。
7. CMFS-closeによってファイルをクローズする。

5 処理時間の解析結果

図2、図3に、それぞれスプリット処理とジョイン処理の実行時間の解析結果を示す。

リレーションのタプル数は512K個（約52万個）、タプル長は256バイト、キー長は4バイトである。従って、1リレーションのサイズは128メガバイトとなる。

キー属性は、それぞれのリレーション内でユニークであり、ランダムに配置されている。両リレーションのキー属性値の分布は同一であり、ジョインによる選択率は100%である。

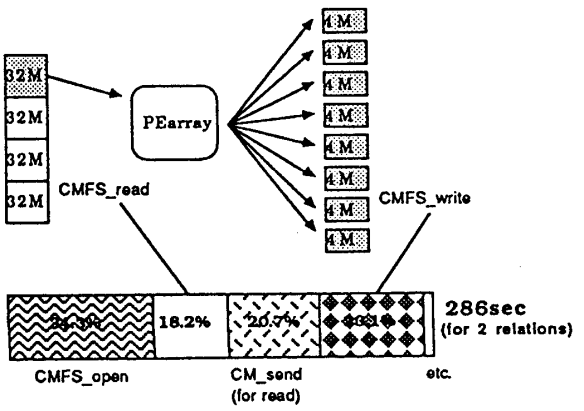

図2: スプリット処理時間の内訳

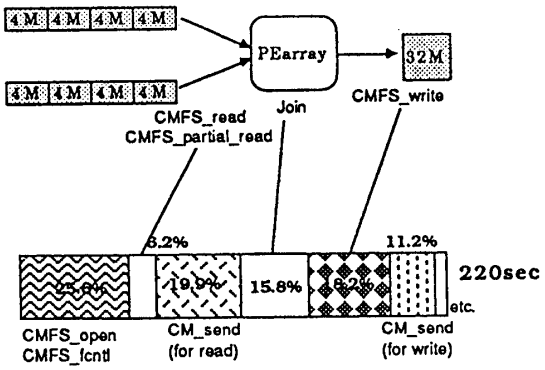

図3: ジョイン処理時間の内訳

CM-read、CM-writeなど、実際に入出力を行なうプリミティブの処理時間の割合が最も大きく、全体の約34%であるが、その他にCM-openと、ストリーミング入出力の際の実質的なオープン処理に相当するCM-fcntlの割合いが約30%と非常に大きいことがわかる。また、CM-sendが全体の約25%の時間を占めているが、これはDataVaultにストアされているデータのVP比と処理する時のVP比が違うため、その変換をするためにかかる時間である。

6 おわりに

本論文では、我々の考案、実装したコネクションマシン上の二次記憶を使った大規模規模関係データベース処理プログラムについて、その処理時間の解析を行なった。

その結果、コネクションマシンのファイル入出力操作ではオープンの処理に非常に時間がかかることがわかった。このことは、ストリーミング入出力を使っても依然解消しない。我々のプログラムの処理時間の内訳を見ても、このために要する時間が最も大きく、コネクションマシンの入出力システムの改良に期待するしかないようである。

しかし、その他に大きな処理時間を占めるCM-sendの処理は、ファイルとして持つ時のVPセットの設定や通信パターンの工夫などによって改善が可能と思われる。また、CM-readやCM-writeなど実際の入出力を行なう部分では、本論文程度の規模の転送量ではあまり高い効率が得られていないので、メモリの実装量が大きくなりより大きな転送量が使えるようになれば、より高い性能が得られるだろう。

今後の課題は、上記の考察をもとに実装の改善を行ない、また、不均一分布など様々な条件下での性能評価を行なっていくことである。

参考文献

[1] 松本 和彦, 喜連川 優, 高木 幹男: 第43回情報処理学会全国大会, 分冊4, pp.279-280, 1991年10月.

[2] 松本 和彦, 喜連川 優: データパラレルソートマージジョインアルゴリズムとコネクションマシンによるその評価, 電子情報通信学会コンピュータシステム研究会, pp.37-44, 1990年10月.

[3] W.Daniel Hillis: The Connection Machine, The MIT Press, 1985.

[4] Thinking Machines Corporation: Connection Machine Technical Summary, 1989.