3R－2

トランスポーズドファイル上での並列結合演算処理方式に関する一考察

武藤 精吾　田村 孝之　中野 美由紀　喜連川 優
東京大学 生産技術研究所

1 はじめに

近年の大規模データベースにおける問合せ処理では大容量のファイルアクセスが頻繁に行われるため入出力に対する負荷が高い。そのため不必要な入出力による性能低下を避けることが必要であり、索引によって必要なタプルだけをアクセスするという工夫が用いられるのが一般的である。しかし、射影演算によってほとんどの属性が捨てられてしまうような問合せにおいては、これとは異なるアプローチが必要となる。リレーションの物理的な格納方式としてトランスポーズドファイルを用いると、タプルを属性ごとに分割することで不要な属性の読み込みを避けることができるためディスクI/Oの減少による性能向上が期待できるが、その反面、タプルを再構成するためには結合処理を繰り返し行う必要がありCPU負荷が高くなる。本報告ではトランスポーズドファイルを用いた並列ハッシュ結合演算の共有メモリ計算機上での実装方式を検討し、その有効性について考察する。

2 トランスポーズドファイル

トランスポーズドファイルとはリレーションのタプルを属性ごとに分割し、独立したファイルに格納したものである。分割前の属性間のつながりはそれぞれにタプルID(TID)を付加することにより維持される。タプルを属性ごとに分割することによって、不必要な属性の読み込みを避けることができ、ディスクI/Oの減少による性能向上が期待できる。トランスポーズドファイルに関する研究は古くから行われていたが[1]、複雑な問合せに対しては属性間の結合処理が高負荷になるため、十分な性能を得ることは困難であった。しかしながら、近年プロセッサの処理性能は著しく向上し、また並列計算機の実用化によって複数のプロセッサを用いて並列処理することが可能となっている。一方、結合演算の高性能アルゴリズムや並列処理による高速化に関する研究も進んでいる。したがってトランスポーズドファイルの問題点となっている属性間の結合処理に並列ハッシュ結合演算方式を適用することで、大規模データベースの問合せ処理に対する性能向上が期待できる。

A Consideration on Parallel Join Processing Method using Transposed Files
Seigo Muto, Takayuki Tamura, Miyuki Nakano and Masaru Kitsuregawa
Institute of Industrial Science, University of Tokyo
Roppongi 7-22-1, Minato, Tokyo, 106 Japan

3 トランスポーズドファイルを用いた並列ハッシュ結合演算実装方式

前述のようにトランスポーズドファイルによる効果を得るためにはCPU負荷の高い属性間の結合処理を高速に行うことが必要である。今回は、並列ハッシュ結合演算を属性間の突き合わせにも応用することでトランスポーズドファイルを用いた問合せ処理の性能向上を目指す。

3.1 並列ハッシュ結合演算

ハッシュ結合演算方式は一方のリレーションの結合属性にハッシュ関数を適用し、ハッシュテーブルを生成するビルドフェイズと、他方のリレーションの結合属性にも同様のハッシュ関数を適用してハッシュテーブルを検索し、結合処理を行うプローブフェイズからなる。ここでは共有メモリ計算機上での並列ハッシュ結合演算方式を考える。

ビルドフェイズにおいて主記憶に読み込まれたタプルに対するハッシュ関数の適用処理は並列に実行することが可能である。このとき、同一ハッシュエントリに対する同時書き込みを防ぐために排他制御が必要である。一般にハッシュエントリの数はプロセッサの数にくらべ極めて多く、そのオーバヘッドは問題にならない。また、プローブフェイズにおいてもタプルの結合属性に対するハッシュ関数の適用、ハッシュエントリの検索、結果の生成等の処理は並列に実行することが可能である。このとき、結合演算結果の書き込みは排他的に行う必要があるが、バッファリングにより出力の単位を大きくすることで衝突を減らすことができる。

3.2 実装方式

今回の実装では分散共有メモリマシンであるHP社Exemplar SPP1000の1つのノードを共有メモリ型の並列計算機として用いる。構成はCPU8台、メモリ512Mバイト、ディスク1台となっている。

ハッシュ結合演算処理はディスクとの読み書きを管理するI/O処理とCPUでのハッシュ関数の適用やタプルの比較、転送などの結合処理の2つに大きく分けられ

る。今回はそれぞれの処理ごとに独立したプロセスを用いて実装することとし、I/O プロセスを 1 台の CPU に割り当て、残りの各 CPU にジョインプロセスを割り当てることにした。

これら 2 つのプロセスは共有メモリ上に設けられたページキューを介してデータの転送を行う。基本的には、ディスク上のすべてのデータを I/O プロセスが読み込んでページ単位で共有ページキューにつなぎ、ジョインプロセスは、データを読み込まれたページがある限り、所定の処理を行うこととなる。今回の実装では、主記憶上に構成される共有のフリーページ管理キューを用意し、I/O プロセスはディスクから 1 ページ分のデータをフリーページ上に読み込み、リードキューに追加する。ジョインプロセスはリードキューから 1 ページ取り、各フェイズに応じた処理をタプルごとに行い、結果を同様にフリーページから獲得したライトバッファに書き込み、いっぱいになるとライトキューに追加する。そして I/O プロセスはライトキューから 1 ページごとにディスクに書き込む。

4 性能評価

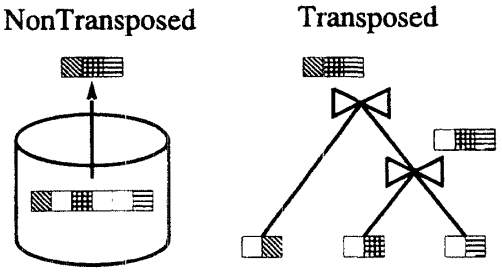

図 1: 性能評価における処理の比較

トランスポーズドファイルを用いて必要な属性のみを読み出して結合処理を行ったときの実行時間とすべての属性をそのまま読み出すときのファイルのリード時間の比較を図 2に示す。

トランスポーズドファイルではタプル数を 524,288 に固定し、タプル長は 4 バイトの TID と 4 バイトの属性の 8 バイトとした。トランスポーズドファイルを用いない場合のタプル長は 100、200 バイトの 2 種類とした。

図 2より 100 バイトのリレーションから 8、16 バイト分の属性を得るときにはそれぞれ CPU が 3、5 台以上のときにトランスポーズドファイルを用いた方が性能がよいことがわかる。24 バイトの場合には CPU が 8 台になってようやく性能が上回ることから、100 バイトのタプル長ではそのうちの 24 バイトくらいまでの属性のみが必要であるときにトランスポーズドファイルが効果的であることがわかる。

リレーションのタプル長が 200 バイトの場合を見てみると、そのうちの 8 バイトが必要なときは CPU の台数にかかわらずトランスポーズドファイルを用いる方が優れた性能を発揮するが、16、24、32、40 バイトと必要な属性の量が増えるにしたがってトランスポーズドファイルを用いた方が優勢となる CPU の最小台数が 4、6、7 と増えて行き、タプル長 200 バイトでは約 40 バイト以内の属性だけが必要とされるときにトランスポーズドファイルを用いるのが有効であることがわかる。

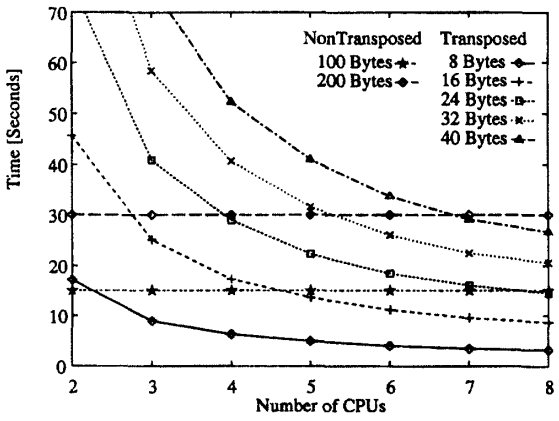

図 2: トランスポーズドファイルを用いた結合演算の実行時間とファイルのリード時間との比較

5 まとめ

今回の報告ではトランスポーズドファイルを用いた並列ハッシュ結合演算の共有メモリ計算機上での実装方式を検討し、測定を行うことによってその有効性について考察を行った。今後は意志決定支援などの大量のデータを扱うシステムの標準的なベンチマークである TPC-D の問合せを用いて測定を行うことにより、実際の問合せにおいてトランスポーズドファイルを用いることによる実行性能について評価する予定である。

参考文献

[1] Don S. Batory, "On Searching Transposed Files", ACM Transactions on Database Systems, Vol. 4, No. 4, December 1979, pp. 531-544.

[2] 喜連川, 津高, 中野, "共有メモリ型マルチプロセッサによる並列ハッシュ結合演算処理とその評価", 情報処理学会論文誌, Vol. 34, No. 5, May 1993, pp. 1019-1030.