大規模 PC クラスタにおけるトランスポーズドファイルを用いた並列関係問合せ処理

4AC－5

田村孝之　小口正人　喜連川 優
東京大学 生産技術研究所

1　はじめに

我々は，近年のパーソナルコンピュータ (PC) の著しい性能向上並びに低価格化と，高速ネットワーク技術の標準化に鑑み，コモディティベースの PC クラスタによる超並列関係データベースサーバの実現可能性を示すことを目的として，100 台の PC を ATM スイッチで結合した大規模 PC クラスタを構築し，並列関係データベース処理系を実装した [1]．各 PC ノードは 200 MHz Pentium Pro を搭載し，64 MB の主記憶と 4.3GB の SCSI ディスクを内蔵しており，155 Mbps の ATM ネットワークで他ノードと相互結合されている．

本システムでは非定型問合せ処理の高速化を目指して I/O 効率の良い並列ハッシュ多重結合演算を実装し，さらにデータ駆動型の実行モデルを用いることで I/O 性能の向上を図っている．100GB TPC-D ベンチマークを用いたこれまでの性能評価により，商用並列 DBMS と比較して高い性能を達成できることが確認された．しかし，データ解析を目的とする問合せでは，アクセスされるアトリビュートはわずかであることが多いので，不要なアトリビュートの読み込みを避けることにより，インデックスが存在しない場合でもデータ転送量を削減することができる．そこで，本稿では各タプルをアトリビュート毎に複数のファイルに分割して格納するトランスポーズドファイルを採用した場合の問合せの処理性能を，TPC-D ベンチマークの問合せに対して詳細に調べたのでその結果を報告する．

2　トランスポーズドファイルによる I/O コストの削減

トランスポーズドファイルは，複数アトリビュートのリレーションを単一アトリビュートから成る複数のファイルに垂直分割する技法であり，特定のアトリビュートに関する集計演算の高速化や，画像データなどの大きなバイナリデータをアトリビュートとするために用いられてきた [2, 3]．現実のデータ解析問合せでは，タプルの一部のアトリビュートしか使われない場合が多いため，トランスポーズドファイルによって不要なアトリビュートに対する無駄な I/O を減らすことで性能向上がもたらされると期待できる．しかし，分離されたアトリビュートから元のタプルを構成するために結合演算を頻繁に実行する必要があり，実行木は非常に複雑になる．

3　TPC-D 問合せによる性能評価

ここでは，標準的な意思決定問合せベンチマークである TPC-D [4] から負荷の高い Query 9 (6 way join) を例に取り，100GB のデータベース に対して実行した時の性能を調べた．ただし，各リレーションはプライマリキーのハッシュ値に基づいて各ノードのディスクに水平分割して格納することとした．また，ファイルアクセスについてはフルスキャンとし，問合せのコンパイル・最適化に要する時間は無視した．

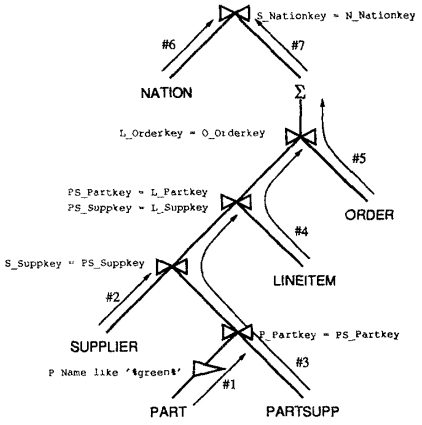

図 1: Bushy tree による Query 9 の実行プラン

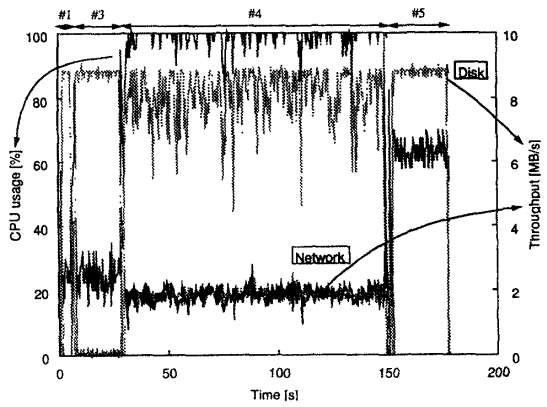

図 2: Bushy tree による Query 9 の実行時トレース

図 1 に通常のリレーション格納方式に対する Query 9 の実行プランを示す．ここでは，PART, SUPPLIER,

---

Parallel relational query processing using transposed files on a large scale PC cluster
T. Tamura, M. Oguchi, and M. Kitsuregawa
Institute of Industrial Science, The University of Tokyo
7-22-1, Roppongi, Minato-ku, Tokyo 106, Japan.

および PARTSUPP 間の結合演算を Right-deep のセグメントとして実行する Bushy tree を用いた. また, Σ の部分は集計演算 (グループ毎の和) を表しており, その後の NATION との結合演算はマスタノードでローカルに行われる.

図 2は実行時の CPU 使用率とディスクおよびネットワーク送受信の実効スループットを測定したものである. フェーズ #4 以外では, CPU 負荷が約 70% 以下であり全体として処理は I/O ボトルネックになっている. フェーズ #4 の LINEITEM による結合演算では CPU 負荷がほぼ 100% を示しているが, ディスクのスループットの低下はわずかであり, I/O ボトルネックと CPU ボトルネックの境界とみなすことができる.

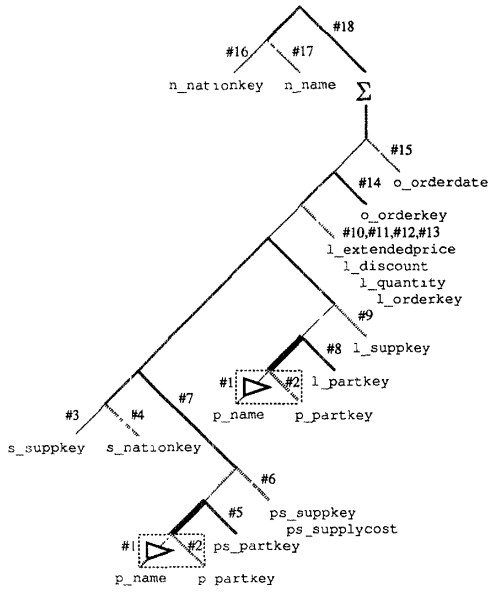

図 3: トランスポーズドファイル上での Query 9 の実行プラン

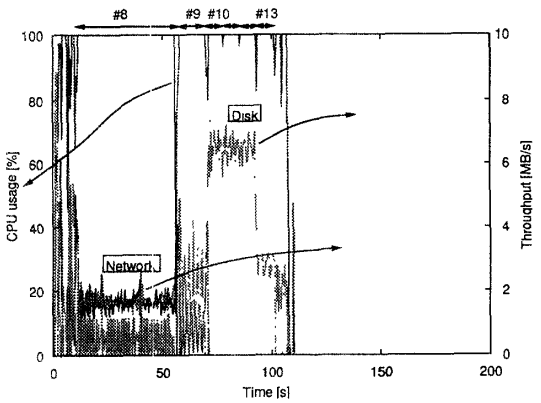

図 4: トランスポーズドファイル上での Query 9 の実行時トレース

トランスポーズドファイルを用いてリレーションを格納した場合の Query 9 の実行プランを図 3 に示す. これは, 基本的に上で述べた Bushy tree に基づき, 分割されたアトリビュートから元のタプルを構成するためのタプル ID 結合を加えたものであるが, LINEITEM の結合時に結合条件として **L_Partkey** と **L_Suppkey** の両方が必要になるため, それぞれについて結合演算を行っている点が大きく異なる. また, **p_name** と **p_partkey** のタプル ID 結合の結果生成されるハッシュテーブルは, **ps_partkey** との結合と **l_partkey** との結合の 2 回に亙って使用される点も特徴的である.

図 4 にトランスポーズドファイル上での Query 9 の実行時トレースを示す. 全体の処理を通じて I/O ボトルネックが CPU ボトルネックに転じ, 実行時間が 40% 程度短縮されている. どのフェーズに於いても CPU 負荷が 100% に近くなり, ディスクスループットが最大値から大幅に低下していることが分かる. 特にそれが顕著に現れているのが, フェーズ #8 の **l_partkey** と **p_partkey** との結合である. このフェーズでは読んだタプルが 100% 選択される上, 結果を元のノードに送り返すためのフィールドが付加されるため, タプル長が長くなって送出される. そのため, 処理負荷は極めて重く, ディスクの実効スループットは 1 MBytes/sec 以下にまで低下してしまう. これを改善するための手段として, **p_name** と **p_partkey** のタプル ID 結合の結果をノード間でハッシュ分割する代わりに, 全てのノードに broadcast するという方法が考えられる. この方法では, **p_partkey** のハッシュテーブルが単一ノードのメモリに収まる必要があり, メモリの利用効率という点では好ましくないが, その結果 **l_partkey** をローカルに結合できるようになり, 処理負荷が軽減されることが予想される. この方式で Query 9 を実行するとフェーズ #8 の時間が大幅に減少し, さらに 30 秒程度の高速化が達成された.

4 まとめ

本論文では, 大規模 PC クラスタ上の並列関係問合せ処理におけるトランスポーズドファイル適用の効果を述べた. TPC-D ベンチマークの問合せに対する性能評価により, 2 倍程度の性能向上が得られることが分かった. これにより, トランスポーズドファイルは, 意思決定アプリケーションに対して非常に有効であることが実証できた.

参考文献

[1] 田村, 小口, 喜連川. ATM 結合 PC クラスタにおける並列関係問合せ処理系の設計と実装. 信学技報 データ工学研究会, Vol. 97, 1997.

[2] D.S. Batory. On searching transposed files. *ACM TODS*, Vol. 4, No. 4, 1979.

[3] P.A. Boncz, W. Quak, and M.L. Kersten. Monet and its geographical extensions: A novel approach to high performance GIS processing. *EDBT*, 1996.

[4] TPC. TPC Benchmark^{TM} D (Decision Support). Standard Specification Revision 1.1, 1995.