3Q－2

並列 SQL サーバ SDC-II における トランスポーズ型ファイル編成適用の検討

田村孝之　喜連川 優　高木幹雄
東京大学 生産技術研究所

1　はじめに

高並列関係データベースサーバ SDC-II (Super Database Computer II) は，データ処理モジュール (DPM) と呼ぶ処理ノード８台を間接多段網 (変形オメガネットワーク) で結合した shared-nothing アーキテクチャを採用しており，さらに各 DPM を４台の磁気ディスク装置と最大７台のプロセッサからなる共有バスクラスタで構成している [1]. SDC-II が目的とするのは大規模関係データベースに対する非定型問合せ処理の高速化であり，I/O 効率が高く，負荷の偏りに強いハッシュ結合アルゴリズムを採用することに加え，インテリジェントな I/O プロセッサとデータ駆動型のプロセス実行モデルを用いることでハードウェアとソフトウェアの両面から I/O 性能自体の向上を図っている.

SDC-II において，これまでに行なわれた性能評価では，リレーションはノード間に水平パーティショニングされており，各タプルを不可分な単位としてファイルに格納することを仮定していた. しかし，データの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，不要なデータ転送に時間を費やすことになりやすい. 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，トランスポーズ型のファイル編成を SDC-II に採用することを検討し，問合せの実行に与える影響を考察する.

2　従来の SDC-II における性能解析

これまで，SDC-II においては，ライトディープ多重結合演算を含むハッシュ結合演算を中心に性能評価を行なってきた [2]. それらの内，ここでは TPC-D ベンチマークに対する多重結合演算を例に取って考えることにする. 図1に 3-way join である Query 3 のライトディープ方式による実行木の例を示す. また，この実行木に従って，SDC-II の１モジュールでスケールファクタ 4 (TPC-D working draft 6.0 における値，現行の仕様での 0.4 に相当) のサイズのデータベースに対して処理を行なった際の実行フェーズ毎の所要時間は表１の通りである.

ここでは，選択アトリビュートに関するインデックス

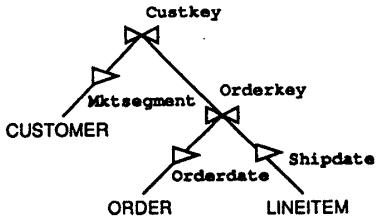

図 1: TPC-D ベンチマーク Query 3 の実行木の例

表 1: Query 3 実行時間の内訳

実行フェーズ	所要時間
ORDER のビルド	8.3 s
CUSTOMER のビルド	1.5 s
LINEITEM によるプローブ	42.1 s
計	51.9 s

が存在しないことを仮定しているので，リレーション全体に対するシーケンシャルリードを行なって全てのタプルを読み込む必要がある. 従って，ディスクから読み込まれるデータのサイズは表２ の中央に示した大きさになる. SDC-II においては，平均データ転送速度が約 2.3 MB/s の磁気ディスク装置を４ 台ストライプ化して用いているため，合計の転送速度は約 9 MB/s であり，処理全体がディスクからのデータ読み込み時間で決定され，完全な I/O バウンドで動作していることが分かる. このような状況下で，デバイスの高速化なしに処理時間のさらなる短縮を図るには読み込むデータ量を減らすしかないが，インデックスの使用は適用できる問合せが限定されるので ad-hoc な問合せを前提とする環境では完全な解とは言えない. [3] では，複数問合せにおけるリレーションのロードを一括して行なうことで，全体の入出力量を減らすことを検討したが，本稿では単一問合せに対しても有効な方法を考えることにする.

表 2: Query 3 におけるリレーションサイズ

リレーション	全ての属性	必要部分のみ
ORDER	67.2 MB	9.6 MB
CUSTOMER	10.8 MB	0.84 MB
LINEITEM	374.4 MB	57.6 MB
計	452.4 MB	68.0 MB

On application of transposed file organization to the parallel SQL server SDC-II
T.Tamura, M.Kitsuregawa, M.Takagi
Institute of Industrial Science, University of Tokyo
7-22-1, Roppongi, Minato-ku, Tokyo 106, Japan.

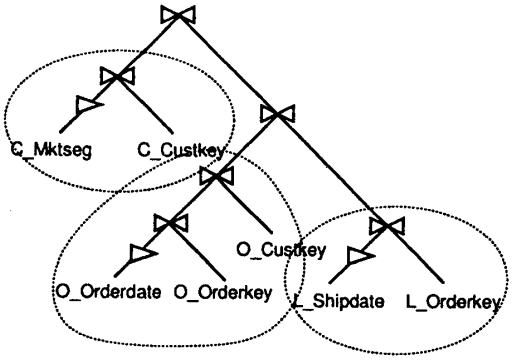

図 2: トランスポーズ型ファイルに対する Query 3 の実行木

そこで, 実際に Query 3 の実行に必要なアトリビュートのサイズだけを取り出してみると表 2の右側のようになり, 約 1/7 のデータしかアクセスされないことが分かる. 従って, リレーションの物理的な格納方式として, それぞれのアトリビュートを独立にアクセスできるようにトランスポーズ型のファイル編成を用いることにすればI/O の効率をさらに高めることができると考えられる.

3 トランスポーズ型ファイル編成への対応

トランスポーズファイルは, データ解析の際の AVG, MIN, MAX などの集計演算を高速に行なうことや, アトリビュートとして大きなバイナリデータを含めるようにすること, 同一アトリビュート値の連続に対して run-length encoding によりデータ圧縮を行なうことなどを目的として用いられてきた技法である [4]. ここでは, フルトランスポーズを考えるため, 単一アトリビュートから成るリレーションがアトリビュートの数だけ作成されることになる. そして, 元のリレーションはそれら全てのトランスポーズ化リレーションの結合演算から導かれる. トランスポーズファイルの欠点として知られているのが更新処理のコストの高さであるが, SDC-II は Decision Support System などの問合せ主体の環境を前提としており, 更新はバッチで行なわれると仮定できるのでこの点は大きな問題にはならない.

問合せを実行する際には, アクセスされるアトリビュートのそれぞれに対応するリレーションの結合演算を行なうことになるが, リレーションの数が増えるため実行木は非常に複雑になる. そのため, Query Optimizer の負担が大きくなるが, 選択・結合述語で参照されるアトリビュートに関しては, 図 2 のようにトランスポーズ化しない時の実行木と同様なものが適していると予測される. それは, この図の点線で囲んだ部分は, それぞれ図 1の 3 つの入力リレーションに対して選択と射影を施したものと同じ結果を生成する部分木であるが, 同一タプルに属するアトリビュート値は同一処理ノード内に格納されるため, これらの結合演算はネットワークを介さずにローカルに処理することができるからである. また, これらの結合演算における結合属性はタプルの ID であり, 効率的な実行が可能である. 述語中に用いられない (`select` リスト中にのみ現れる) アトリビュートに関しては, 上の実行木の出力と結合することで, 中間的なタプルのサイズを最小限にとどめることができる.

表 3: データ量一定の時の結合演算におけるタプル長と処理時間の関係 (GRACE Hash Join, 処理モジュール数 4)

タプル長	タプル数	均一負荷	不均一負荷
104 B	2 M	57.52 s	138.25 s
208 B	1 M	54.76 s	109.59 s
416 B	0.5 M	54.45 s	107.40 s

以上のような実行方式によって, 図 1で読み捨てられていたアトリビュートに対する I/O が減るため, 大幅な性能向上が期待されるが, このように非常に短いタプルに対する結合演算においては, 表 3 に示すように CPU 側がボトルネックとなって単純に I/O だけで性能を見積もることはできなくなってしまうことに注意しなければならない. 完全なトランスポーズ化による結合演算のオーバヘッドが, 入出力量の減少による効果を上回ってしまった場合には, 同時にアクセスされることの多いアトリビュートについては同じファイルに格納するなどの部分的なトランスポーズ化の検討も必要となる.

4 まとめ

SDC-II における問合せ処理をさらに高速化するために, トランスポーズ型ファイル編成を導入して入出力量の減少を図ることを検討した. 今後は SDC-II 上での実装を行ない, 実質的な性能向上について評価を行なう予定である.

参考文献

[1] 中村, 平野, 田村, 喜連川, 高木. “スーパーデータベースコンピュータ SDC-II におけるシステムソフトウエアの設計と実装”, 信学論 Vol.J78-D-I, No.2, 1995, pp.129–141.

[2] 中村, 田村, 喜連川, 高木. “スーパーデータベースコンピュータ SDC2 における多重結合演算の実装と評価”, 信学研究会 CPSY94-28, 1994.

[3] 田村, 喜連川, 高木. “並列 SQL サーバ SDC-II におけるバッチ問合せ処理方式”, 情処全国大会 51(4), 1995.

[4] D.S.Batory. “On Searching Transposed Files”, ACM TODS, 4(4), 1979.