5H-5 Symmetry S81 におけるGRACE HASH方式の実装と評価

津高 新一郎 中野 美由紀 喜連川 優 高木 幹雄
東京大学 生産技術研究所

1 はじめに

関係データベース処理の中で、結合演算は、選択演算などの他の関係演算に比べて処理負荷が重いことは良く知られており、その処理負荷を軽減すべく今まで種々の結合演算処理方式が提案されてきた[1,2,3,4,5]。なかでもハッシュ操作に基づく結合演算処理は、従来のソート処理に基づく方法に比べて高い性能が得られ、数々の研究成果が発表されている。

一方、データベースマシンの分野では、処理性能の向上を目指し、並列処理技法を取り入れたアーキテクチャが多数考案されている。当研究室では、共有メモリマルチプロセッサマシン上での関係演算処理方式として、最も処理負荷の重い結合演算の実装について検討している。32MBの共有メモリ、18台のプロセッサ、4台のディスクから構成されるSequent社のSymmetryを用い、この上でGRACE HASH方式を用いた結合演算の実装を行なった。本稿ではその実装方式と評価について述べる。

2 GRACE HASH結合演算処理方式

ハッシュ操作に基づいた結合演算処理とは、演算処理対象を独立した複数の空間に分割し、結合処理に必要なデータ検索空間を狭めることで、演算処理の高速化を図る方法である。以下、GRACE HASH結合演算方式の処理の流れについて述べる。本方式は、次のような2つのフェイズから構成される。(結合演算の対象となるリレーションをR、Sとする。)

分割フェイズ リレーションRの各タプルの結合属性にスプリット関数を適用してバケットに分割し、中間リレーションRiに書き出す。続いて、リレーションSも同様にしてバケット分割する。

結合フェイズ 各バケット毎に結合演算処理を実行する。異なるスプリット値を持つバケット同士は結合される可能性が無いため、等しいスプリット値をとるバケット同士が主記憶上にステージングされて結合処理が施される。

3 GRACE HASH方式の実装

当研究室のSymmetry S81は、32MBの共有メモリ、18台のプロセッサ(i386、16MHz)4台のディスクから構成される。4台のディスクは各々別のチャネルに接続され、並列にアクセスすることが可能である。

GRACE HASH方式を本マシン上に実装する際には、共有メモリ上のデータに対するマルチプロセッサによる並列処理効果を考慮すると共に、入出力処理に対する並列処理方式についても留意しなければならない。そこで、今回の実装方式の特徴を以下に示す。

- 分割フェイズにおけるスプリット関数によるバケット分割、あるいは、結合フェイズ時の結合処理等のデータ処理を行なうプロセッサ群とディスクの入出力アクセス処理を行なうプロセッサ群を完全に分離し、各々の並列処理効果を最大限に得る。
- リレーションを複数台のディスクに分割して置き、並列アクセスによる入出力の高速化を図る。
- 入力用のプロセッサ、出力用のプロセッサとしてディスク1台につき各々1台のプロセッサを割り当て、入出力管理を専用プロセッサで集中して行なう。
- 入出力バッファをダブルバッファ化し、一方のバッファをディスクとのデータ転送用、もう一方をプロセッサとのデータ転送用とすることで、プロセッサの入出力待ち時間を減らす。
- 各バケット毎の結合処理では、ハッシュ関数を用いて小さなデータクラスタ群を生成し、プロセッサが並列に各データクラスタ毎に処理を行なう。

以下に、GRACE HASH方式の実装について各フェイズ毎に詳述する。但し、結合演算の対象となるリレーションをR、S、入力用プロセッサをPi, 出力用プロセッサをPo, その他のプロセッサをPxとし、ディスクの台数をd台とする。

分割フェイズ(図1)

1. ディスク上のリレーションRをd個のプロセッサPiが読み出し、入力用ダブルバッファに格納する。
2. n個のプロセッサPxは入力バッファ中のタプルを並列にローカルメモリにコピーした後、結合属性にスプリット関数を適用し、その値に応じた出力用ダブルバッファにコピーする。この時バッファがいっぱいになると、書き出し要求メッセージを出すと同時にバッファの切り替えを行う。
3. d個のプロセッサPoは書き出し要求メッセージを受けて、出力用ダブルバッファの内容を部分リレーションRjとしてディスクに書き出す。
4. リレーションSに関しても同様に1.~3.の処理を行なう。

結合フェイズ(図2、3)

1. ディスク上の部分リレーションRjをPiが読み出し、入力用ダブルバッファに格納する。
2. Pxは入力バッファ中のタプルをローカルメモリにコピーしたのち、結合属性にハッシュ関数を適用し、その値に応じた共有メモリ領域にコピーする。
3. 1.~2.の処理をRjのタプル全てについて行なう。
4. 次に、Rjに対応する部分リレーションSjをPiが読み出し、入力用ダブルバッファに格納する。
5. 各Pxは入力バッファ中のタプルを並列にローカルメモリにコピーしたのち、結合属性にハッシュ関数を適用し、その値に応じた共有メモリ領域を参照する。結合属性値が一致すると、両方のタプルは出力用ダブルバッファにコピーされる。
6. Poは出力バッファを監視しており、いっぱいになった時点で結果リレーションとしてディスクに書き出す。
7. 1.~6.の処理を全ての部分リレーションRj、Sjについて行なう。

4 性能評価

タプル長208バイト、結合属性長4バイト、属性値がユニークで順番がランダムであるようなリレーションに対し、4kバイトのページ256個からなる1Mバイトのメインメモリをステージングバッファとして用いてコストの測定を行なった。

まず、入出力以外に用いるプロセッサの数が1台の時に、ディスクの台数とリレーションのサイズの変化させてコストを測定した。その結果を図4.に示す。処理コストはリレーションの件数に比例して増加しており、Symmetry上で我々の提案したGRACE HASH実装方式が有効であることが確認できる。また、ディスクの並列アクセスに関しては台数を増やしてもほぼ処理コストはリレーションの件数に比例して増加しており、GRACE HASH方式の性能として理想的な結果が得られている。

---

[0]The Implementation and Evaluation of the GRACE HASH Algorithm on Symmetry S81
S.Tsudaka, M.Nakano, M.Kitsuregawa, M.Takagi
Institute of Industrial Science, University of Tokyo

また、リレーションの大きさを2万件とし、ディスクの台数と入出力以外の処理に用いるプロセッサの数を変化させた。その結果を図5.に示す。この図から、ディスクによる台数効果がみられ、ディスクの台数の増加に従って処理性能はほぼ線形に伸びている。しかし、プロセッサが9台の時、ディスク2台では約1.6倍、ディスク4台では約3倍しか処理性能が向上していないことがわかる。これは、リレーションをディスク毎に分割した結果、各ディスクをアクセスする入出力プロセッサ間で同期をとる必要が生じ、オーバーヘッドが生じるためであると思われる。また、プロセッサが1台の場合、9台の時と比べてディスクの台数効果はあまり顕著でない。これは、ディスクの複数化によって入出力コストが軽減されると、入出力以外のコストが支配的になってくるためと思われる。

さらに、共有メモリ上のデータ処理に対するマルチプロセッサの並列処理効果を明確にするため、リレーションを1万件とし、ディスクとのデータ転送を行なわない場合のハッシュを用いた結合処理フェーズの性能測定を行なった。その結果を図6.に示す。プロセッサの増加に従い、処理性能はほぼ線形に伸びていることが確認でき、Symmetry上での並列処理が結合処理コストの軽減に効果的なことがわかる。

5 おわりに

GRACE HASH方式のSymmetry上における実装方法とその性能について報告し、ハッシュを用いた処理方式が有効であることが確認された。関係データベース演算の中でも最も負荷の重い結合演算に対して、ディスクの台数効果、プロセッサの台数効果ともに明確に認められ、Symmetryが関係データベースシステムを構築するうえで極めて高い性能を持っていることが確認された。

現時点ではディスクとのデータの転送にシステムコールを使っているため、入出力はOSの介在によりバッファリング等の管理を受け、そのコストを詳細に評価するのは困難である。従って、OSによる管理を受けないローデバイスを使用した実装方式を現在考慮中である。また、結合演算だけでなく、集計演算等のハッシュを用いた処理方式が有効な関係データベース演算についてもSymmetry上で検討を行なう予定である。

参考文献

[1] 喜連川優, 他.「動的処理バケット選択方式による結合演算処理の詳細評価」, 情処学会論文誌第30巻,1989.

[2] 喜連川優, 他.「動的処理バケット選択手法に基づくハッシュ結合処理方式とその性能評価」, 情処学会論文誌第30巻, 1989.

[3] M.Kitsuregawa,et al. "The Effect of Bucket Size Tuning in the Dynamic Hybrid GRACE Hash Join Method", VLDB 89,1989.

[4] D.J.Dewitt,R.Gerber. "Multiprocessor Hash-Based Join Algorithms", VLDB 85,1985.

[5] L.D.Shapiro. "Join Processing in Database Systems With Large Memories", ACM TODS,Vol.11,No.3,1986.

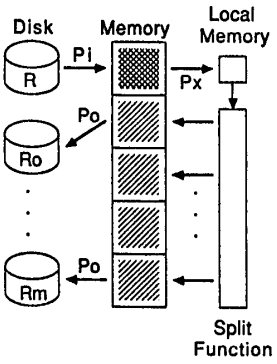

図 1: 分割フェイズ

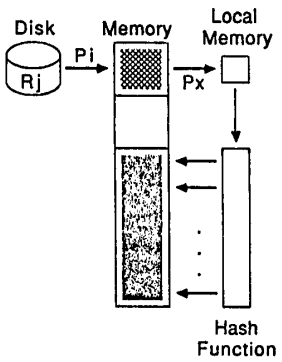

図 2: 結合フェイズ－1

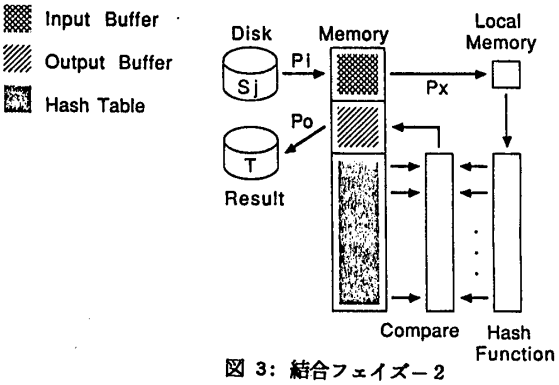

図 3: 結合フェイズ－2

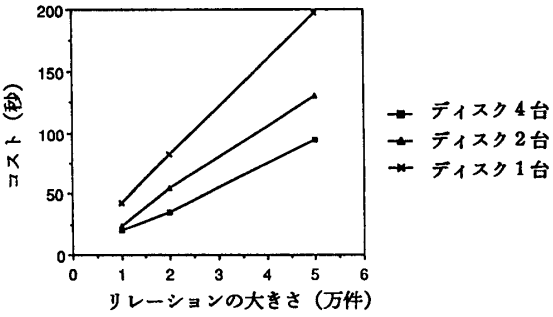

図 4: 測定結果－1

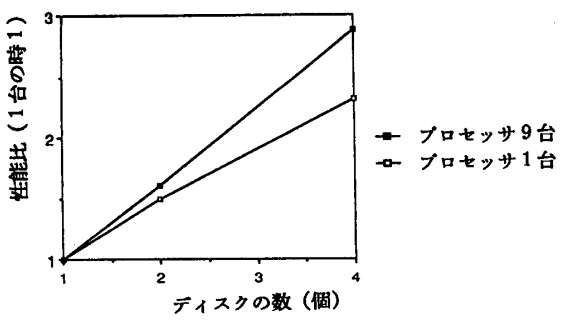

図 5: 測定結果－2

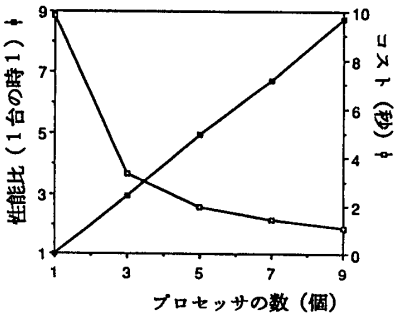

図 6: 測定結果－3