分散共有メモリ並列計算機における関係結合演算の性能

4AC－4

中野 美由紀, 今井 洋臣†, 喜連川 優
東京大学生産技術研究所

1 はじめに

近年、情報化社会の急激な成長に伴い、生成、収集されるデータは増大する一方であり、これらのデータを効率良く管理、運営するため大規模データベースシステムでは肥大化するデータへの容易な拡張性、複雑化する問合せへの迅速な応答に対応する高性能化を求め、並列処理化が急速に進められている。しかしながら、共有メモリ型または分散メモリ型並列計算機における関係データベース処理の研究は多く行われている [1] が、分散共有メモリ型 (DSM) 並列計算機上での並列処理方式、性能解析の研究は今だ少ない [2, 3]。現状のDSM 並列計算機の性能評価は、数値計算に見られるようなアクセス局所性の顕著なアプリケーションを対象とした解析が主であり、データベース処理のような広大な主記憶空間をランダムアクセスする場合にはそれらの解析結果は適応できない。本報告では、関係データベースでも負荷の高い結合演算を対象として、Cache Coherent Nonuniform Memory Architecture (CC-NUMA) の DSM 並列計算機上の並列データベース処理におけるバッファアクセス方針が処理性能に与える影響について、ハッシュ結合演算を対象とした性能解析を行う。

2 分散共有メモリ並列計算機 SPP 1600 のメモリアクセス特性

CC-NUMA DSM 並列計算機 HP Exemplar SPP 1600 は最大 8 CPU (PA-RISC 7200, 120MHz) をクロスバスイッチにより密結合し一つのノードを構成し，各ノード間を SCI プロトコル [4] を用いた高速ネットワークにより結合することにより、キャッシュコヒーレントなメモリ分散共有機構を 64byte のキャッシュライン単位で実現している．本報告では、1 ノード当たり 8CPU,512MB メモリの 4 ノード構成、全体では 32CPU と 2GB のメモリからなるシステム構成を用いた。

Mapping	Access	Invalidation	Fault	Lock	cost(usec)
Local	read	No	No	No	0.8
Local	read	Yes(Invalidated)	No	No	5.1
Local	write	No	No	No	0.8
Local	write	Yes(Invalidate)	No	No	5.8
Local	write	No	Yes	No	5.1
Local	write	No	No	Yes	6.0
Local	write	Yes	No	Yes	16.8

Mapping	Access	Invalidation	Fault	Lock	Cache hit	cost(usec)
Remote	read	No	No	No	Yes	0.8
Remote	read	No	No	No	No	4.0
Remote	read	Yes	No	No	No	4.6
Remote	write	No	No	No	Yes	2.0
Remote	write	No	No	No	No	4.0
Remote	write	Yes	No	No	No	5.8
Remote	write	No	Yes	No	No	4.6
Remote	write	No	No	Yes	Yes	12.2
Remote	write	No	No	Yes	No	14.5

表 1: SPP 1600 におけるメモリアクセス性能特性

SPP 1600 上のメモリクラスごとのアクセス特性について表に示す。単純なアクセスプログラムにより、ローカルおよびグローバルメモリに対する read,write コスト、リモートアクセス時間、これらのメモリアクセスに伴うコヒーレンシにおける invalidation のペナルティ等を調べ、以下の表 1に示した。メモリアクセスは、4 バイト整数の読み書きをキャッシュライン (64 バイト) おきに 64MB 連続なアドレス空間上で行い、cost として、1 キャッシュラインのアクセス時間を示した。Mapping は、アクセス対象となる空間がアクセスするノードからみて、自ノード内メモリに割り当てられているか (Local)、他ノードのメモリに割り当てられているか (Remote) を示す。Invalidataion, Fault はアクセス時のデータ領域の状態を示す。Invalidation は、read 時にアクセス対象となる領域が他ノードからすでに書き込みが行われていること意味し、write 時には他ノードに参照されていることを意味する。Fault は、write 時にアクセス対象となる領域が他ノードによりすでに書き込みが行われていることを意味する。Lock は、アクセス前に各ノード間で lock 同期をとる場合の書き込み時間、つまり、lock 命令から unlock 命令までの時間を意味する。cache hit は、remote アクセス時に、アクセス対象がそのノード上のキャッシュにあるか否かを示す。

3 分散共有メモリ並列計算機におけるハッシュ結合演算処理方式

3.1 並列ハッシュ結合演算処理モデル

DSM 並列計算機におけるハッシュ結合演算処理モデルを図 1に示す。ハッシュ結合処理は、ビルドおよびプローブフェーズに分けられ、それぞれのフェーズでは、ハッシュテーブル、ビルドリレーションおよびプローブリレーション用データバッファが主としてアクセスされる 3 つのデータ領域であり、入出力、ビルド処理、プローブ処理プロセスからそれぞれの領域に対し、アクセスが行われる。

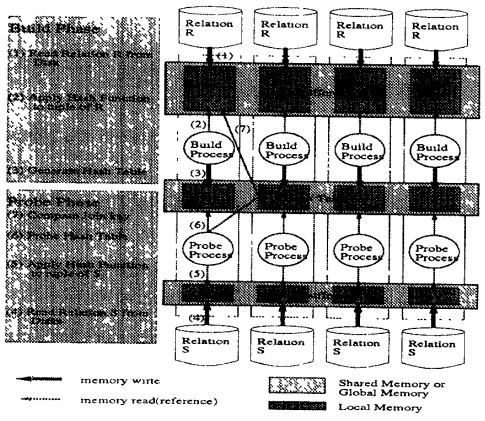

図 1: Parallel Hash Join Model on DSM Environment

Performance Analysis of Hash Join Processing in Distributed Shared Memory Machines
Miyuki NAKANAO, Hiroomi IMAI and Masaru KITSUREGAWA
Institute of Industrial Science, University of Tokyo
roppongi 7-22-1, Minato-ku, Tokyo, 106 JAPAN
† 現ソニー アーキテクチャ研究所

3.2 バッファ領域の獲得方法、アクセスコストによる処理方式の分類

上述の3つのデータ領域および3プロセスから構成される並列ハッシュ結合演算処理モデルの上で表1におけるメモリアクセス特性を考慮し、DSM 並列計算機におけるハッシュ結合演算方式について考察する。バッファの獲得に関しては、ハッシュテーブルをグローバル領域またはローカルに獲得した場合およびデータバッファをグローバル領域およびローカルに獲得した場合が考えられる。さらに、領域の割り付け方針が決定すると、各プロセスからのアクセス方針も表1から他ノード上の参照、書き込みあるいは Invalidation の発生を避けるようにすることで、表2に示す4方式 Shared Everything (SE) 方式, Shared Hash Table (SHT) 方式, Local Hash Table (LHT) 方式, LHT with Remote reference (LHT-R) 方式に定まる。 まず、データバッファに関しては、そ

	Strategy	Shared E	Shared HT	Local HT	LHT- R
	Data Buffer	Global	Local	Local	Local
	Hash Table	Global	Global	Local	Local
I/O process	Build Phase(1) Data Write	inter node	intra node	intra node	intra node
Build process	Build Phase(2) Data Reference	inter node	intra node	inter node	inter node
Build process	Build Phase(3) Hash Table Generation	inter node	inter node	intra node	intra node
I/O process	Probe Phase(4) Data Write	inter node	intra node	intra node	intra node
Probe process	Probe Phase(5) Data Reference	inter node	intra node	inter node	intra node
Probe process	Probe Phase(6) Hash Table Reference	inter node	inter node	intra node	inter node
Probe Process	Probe Phase(7) Build Data Reference	inter node	inter node	intra node	inter node

表 2: Four Buffer Management Strategies

のアクセスは主として図1における Build Phase (1) と Probe Phase (4) において入出力プロセスのディスクからのリレーションの読み込み時における書き込みとその後データ参照が考えられる。分散共有メモリ環境において、通常ディスクは個々のノードごとに直接つながっていることが多い。また、入出力プロセスは基本的にデータバッファに向かって書き込みのみである。したがって、リレーションを読み込むバッファは、グローバルな領域に獲得するよりは、個々のローカルノードごとにバッファを獲得して書き込む方が1からわかるように効率がよい。そこで、SE 方式は単純な共有メモリ型計算機の実装を模してデータバッファをグローバルに獲得しているが、他の方式ではノードローカルに獲得するものとする。ハッシュテーブル領域に関してもデータバッファと同様にビルドフェーズ時に各ノードごとにローカルに生成することが可能である。そこで、ハッシュテーブルをグローバルかつデータバッファをローカルに獲得 (SHT 方式)、または、ハッシュテーブルおよびデータバッファともにローカルに獲得 (LHT & LHT-R 方式) の2つの組合せがある。さらに、プロセスからのメモリアクセスという観点から、プローブフェーズ時は、入出力プロセスが書き込んだデータバッファを他ノード上のプローブプロセスが参照し、自ノード上のハッシュテーブルを参照する (LHT 方式) か、プローブプロセスはデータバッファは自ノードのみを参照し、他ノードのハッシュテーブルを参照する (LHT-R 方式) こととなる。

4 性能解析

今回の測定では、それぞれのノード毎に入出力プロセス、ビルドプロセス、プローブプロセスを1 CPU ずつに割り当て、4ノードを用いて測定した。それぞれのノードのキャッシュはあらかじめグローバル領域をランダムにアクセスすることで初期化されている。以下の結果では、4方式のメモリアクセス時間の差を明確にするために、いずれの方式でも同じだけ必要となるディスクアクセス時間は除いている. 図

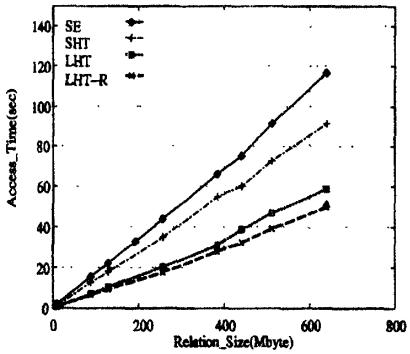

図 2: 処理時間

2からわかるように、それぞれの方式により、大きく性能が異なる。特にメモリアクセス特性を考慮せず、単純に共有メモリ型計算機の実装を模した SE 方式では、十分な処理性能が出ないことが確認できる。また、最も性能の良い LHT-R 方式が SE 方式と比較して、約 58% 以上の性能向上がみられる。ハッシュテーブルをグローバル領域に確保する SE, SHT 方式よりも、ハッシュテーブルを各ノードごとに分割して割り付ける LHT, LHT-R 方式が性能がよいのは、主として lock のオーバヘッドおよびデータバッファをノード毎にアクセスし、他ノードからの書き込みを避けることによる。また、SE 方式と SHT 方式の差は、データバッファ領域の書き込みを他ノードの領域へも書き込むか自ノード内のみとするかの違いによる。さらに、プローブフェーズにおけるデータバッファへ書き込み時の invalidation を避けた LHT-R 方式が LHT 方式よりも性能が向上していることが確認できる。

5 まとめ

本研究では、DSM 並列計算機上の並列データベース処理方式の実装を行うために、並列ハッシュ結合演算を対象として分散共有メモリアーキテクチャに特有の問題に対して、性能解析を行った。上述のように、DSM 並列計算機では単純な実装方式では共有メモリへのランダムなアクセスによりメモリ一貫性処理の頻度が高くなり、処理性能が下がるという問題点がある。バッファ領域獲得時にそれぞれのバッファのアクセスを考慮してノード毎に局在化させることにより、メモリ一貫性のためのトラヒックを下げ、大きく性能を向上させることを明らかにした。

参考文献

[1] P.Valduriez : *Parllel Database Systems: open problems and nes issues*, Int. Journal on DPDS, Vo.1, No.2, pp.(1993)
[2] A.Shardal and J.F.Naughton: *Using Shared Virtual Memory for Parallel Join Processing*, Proc. of SIGMOD '93, pp.119-128, 1993
[3] L.Bouganim, D.Florescu and P.Valduriez: *Dynamic Load Balancing in Hierarchical Paralle Database Systems*, Proc. of 96 VLDB, pp.436-447(1996)
[4] Nagi M.Aboulenein,Stein Gjessing,James: R.Goodman and Philip J.Woest: "Hardware Support for Synchronization in the Scalable Coherent Interface(SCI)".