1R－8

分散共有メモリ計算機上における並列ハッシュ結合演算処理方式の一考察

今井 洋臣, 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，共有メモリ計算機上での並列関係データベースシステムの商用化が著しい．しかし共有メモリ計算機は技術的な限界から，数十プロセッサ程度の並列度しか得られない．一方，分散メモリ計算機は高いスケーラビリティが得られるが，メッセージパッシングによる並列プログラミングが必要となり，並列処理の実装が難しい．そこで，アドレス空間としては共有を可能としながら，高いスケーラビリティが望める分散共有メモリ計算機が最近着目されている．

データベース処理では対象となるデータ量は増大する一方であり，常に性能向上が求められている．従って，拡張性の高くかつ並列処理実装の容易な分散共有メモリアーキテクチャは並列データベースシステムにおいて重要なプラットフォームと思われる．しかしながら，[1] にあるように，メモリの局所性を意識せず，単純な共有メモリ型処理方式を実装すると，各ノード間でスラッシングがおき，性能の低下を招く．また，分散共有メモリ計算機上に適した並列データベース処理方式についての検討はほとんど行なわれていない．

本稿では，並列ハッシュ結合演算を用い，分散共有メモリ計算機，特に Cache-Coherent NUMA における並列データベース処理実装方式について考察を行なう．商用計算機 (CONVEX 社製 Exemplar SPP1000) 上に，共有メモリ型実装方式と我々が提案するメモリのアクセスの局所性を意識した方式を実装し，その結果を比較検討する．

2 Exemplar SPP1000

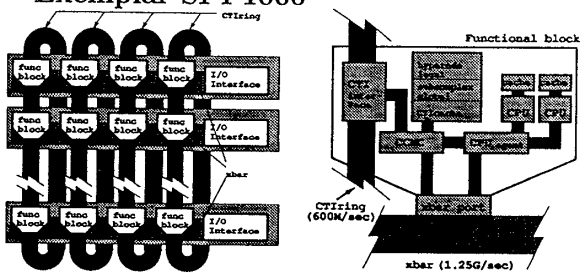

図 1: Exemplar SPP1000 の構成

Exemplar SPP1000 は分散共有メモリ型計算機である．2 つの CPU，CPU エージェント，メモリユニット，CCMC (choherent memory controller) が一つのブロックを構成する．このブロック 4 つを 5x5Crossbar(1.25GB/sec) というバスで密結合してノードを構成する．ノード内は共有メモリである．さらにノード間は CTI-ring(600M/sec) というネットワークリングで疎結合されている．この CTI リングにより分散共有メモリが実現されており，他のノードのメモリ空間がノード内のキャッシュ空間 (CTI キャッシュ) にマッピングされる．SPP100 のメモリ空間は大きく以下の三つのクラスとしてユーザからアクセスできるが，キャッシュそのものは意識されない．

1. 他のノードからは参照できないローカルメモリ
2. 他のノードからも参照可能なローカルメモリ
3. 他ノードのメモリ空間を共有するグローバルメモリ

本研究室のマシンは 4 ノード構成で，32CPU と 1GB のメモリを持っている (8CPU+256MB/node)．

3 並列ハッシュ結合演算実装方式

関係データベース処理の結合演算処理は非常に負荷の高い処理であり，並列処理による高速化の研究が多く行われている．今回の実装では並列 Grace ハッシュ結合演算を対象として単純な共有メモリ型処理方式 (以下 SE 方式) とメモリの物理的な分散を意識しアクセスの局所性を考慮した処理方式 (以下 SN 方式) の 2 つの方式を実装した．並列 Grace ハッシュ結合演算処理はリレーション R からハッシュテーブルを生成するビルドフェーズ及びそのハッシュテーブルを走査しリレーション S との結合処理を行うプローブフェーズからなる．本実装では両方式ともリレーションを読み出すリードプロセス，データバッファ内のタプルをハッシュテーブルに登録するビルドプロセス，ハッシュテーブルを走査するプローブプロセスの三つのプロセスから構成される．

ビルドフェーズ 各ノードの異なるプロセッサ上でリードプロセスとビルドプロセスが並列に動作する．

SE 方式ではメモリのローカリティは意識しない．従ってハッシュテーブル領域，データ領域共にグローバルメモリ上に獲得し，データ領域はバッファ単位にリスト形式で管理する．各ノード上のリードプロセスはリレーションをバッファに書き込む．各ノード上のビルドプロセスは書き込まれたバッファを一つずつ取り込み，各タプルごとにハッシュテーブルに登録する．

SN 方式ではハッシュテーブル領域を各ノード内のローカルメモリ上に獲得する．またデータ領域もノード毎に他ノードから参照可能なローカルメモリ上に獲得し，バッファ単位にリスト形式で管理する．SN 方式ではノード毎にハッシュテーブルが分割されるため，リードプロセスではリレーションを自ノード内のバッファに書き込む時点で該当するハッシュテーブルを持つノードのバッファに振り分ける．バッファは一杯になると該当するノードのバッファリストにつながれる．ビルドプロセスは各ノード内のバッファリストにあるバッファを取り込み，ローカルメモリ上のハッシュテーブルに格納する．

プローブフェーズ 各ノード上の異なるプロセッサ上でリードプロセスとプローブプロセスが並列に動作する。ビルドフェーズと同様にプローブリレーションを読み込むバッファ

Consideration on Parallel Hashjoin in Distributed Shared Memory Architecture
Hiroomi IMAI, Miyuki NAKANAO and Masaru KITSUREGAWA
Institute of Industrial Science, University of Tokyo
roppongi 7-22-1, Minato-ku, Tokyo, 106 JAPAN

をSE方式ではグローバルメモリ上に獲得し、SN方式では各ノード上のメモリ上に獲得する。しかし、ビルドフェーズと異なり、ハッシュテーブルを走査し結合処理を行った後はプローブリレーションを保持する必要はないため、数個のバッファしか用意されていない。

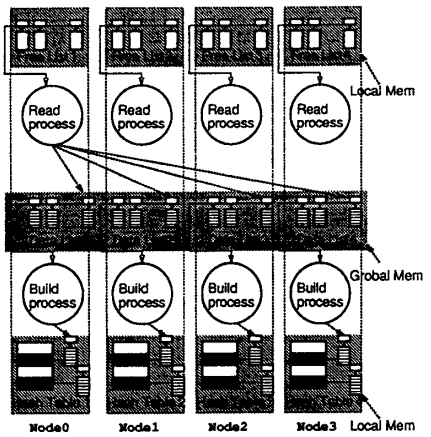

図 2: SN方式におけるビルドフェーズ

4 実装結果と考察

4.1 処理時間

SE, SN方式におけるデータサイズを変化させた場合のビルドフェーズ, プローブフェーズ, 合計処理時間の結果を図3,4に示す. 本測定では4ノードを用い, データサイズを2MB～128MBまで変化させた. またここではメモリアクセスストラテジーの差を確認することが目的であるため, ディスクアクセス時間は除いている. 全体の処理時間からSN方式はSE方式に比べ40%以上の性能向上がみられ, メモリの物理的な分散を意識してプログラミングすることは効果があるといえる.

ビルドフェーズの時間に関してはSN方式がSE方式より約45%程度の性能向上がみられる. SE方式ではどのノード上のメモリかを意識せずにデータの書き込みを行うため, 他のノードのメモリへの書き込みが生じる. 一方SN方式では書き込みはローカルメモリに限り, 他のノードのメモリ空間は参照のみである. つまり, データを分割によってメモリアクセスを局所化し, 負荷の高い他ノードメモリ空間への書き込みを避けることが分散共有メモリにおいて性能向上の大きな要因となる. プローブフェーズにおいても同様にSN方式がSE方式に比較して性能向上がみられる.

SE, SN方式共にビルドフェーズとプローブフェーズは同じサイズのデータアクセスを行っているが, プローブフェーズの処理時間が早い. これはビルドフェーズが書き込んだ全てのページをハッシュテーブルのボディとして保持する必要があるのに対しプローブフェーズでは数個のバッファのみを用いているため, キャッシュエントリ用のディレクトリ作成などのオーバヘッドが少ないためである.

4.2 台数効果

SN方式のビルド, プローブの合計時間の台数効果を図5に示す. SN方式ではノードが1台から2台に増えたところで台数効果が1.7倍程度になり, 2台から4台までは線形に台数効果が得られる. これはノードが1台の場合は全てがノード内ローカルメモリへのアクセスだが, ノードが2台以上の場合の他のノードのメモリ空間の参照があるためである. したがってノード2台以上の場合の結果を用いれば台数効果は効いている. またSE方式の結果も参考として示す. 同様にSE方式でノード一台の場合, アクセスするページを全て自分のCTIキャッシュに取り込んでしまうため, SN方式と同等の性能になる. しかし2台目以降では他ノードのメモリ空間への書き込みが生じるため性能が下がる.

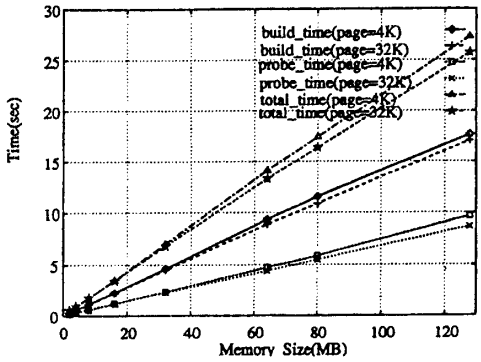

図 3: SE方式 処理時間 (4ノード)

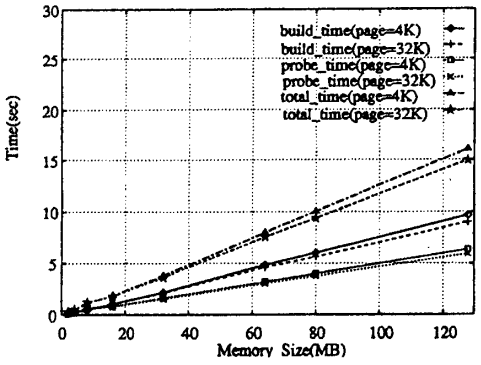

図 4: SN方式 処理時間 (4ノード)

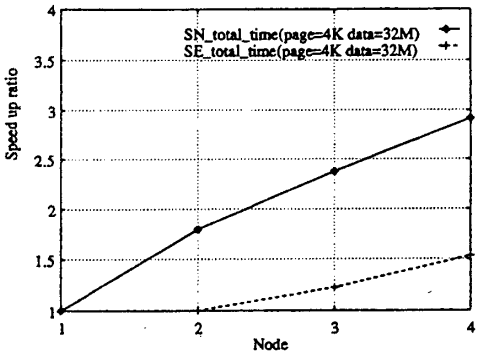

図 5: 台数効果

5 終わりに

本報告では分散共有メモリ計算機上での並列データベース処理を単純な共有メモリ型方式とメモリの分散を意識しアクセスの局所性を考慮した方式とで実装し, メモリの分散を意識したプログラミングの有効性を示した. 今後ノード数が多くなった場合のSE, SN方式の性能評価, データにスキューがある場合のローカリティを意識したアルゴリズムの検討等を行いたい.

参考文献

[1] A.Shardal and J.F.Naughton: *Using Shared Virtual Memory for Parallel Join Processing*, Proc. of SIGMOD '93, pp.119-128, 1993