4Q-2

Hypercubeマシン上でのJOIN演算
- 最適配置の効果 -

平野 聡　楊 維康　喜連川 優　高木 幹雄
東京大学 生産技術研究所

1 概要

データベースマシンのアーキテクチャの並列化の研究が進展するに伴い、負荷の重い結合演算の高速化アルゴリズムの開発が必須となっている。我々は[1]において、ハイパーキューブ・アーキテクチャのデータベースマシンのためのレコード最適配置アルゴリズムを示した。これはハイパーキューブのようなネットワーク結合では不可避のネットワーク・オーバーヘッドを、平均転送距離を小さくすることにより削減することを目的とするものである。本論文では精密なシミュレーションにより最適配置の効果を明らかにした。ネットワークの転送コストがディスクの転送コストを上回り、ネットワークのオーバヘッドが大きい場合、高い効果が得られることがわかった。

2 ジョイン演算とリレーションの最適配置

結合演算のアルゴリズムはGRACE hash方式[2]を仮定した。GRACE hashでは結合演算の対象のリレーションをディスクから読み出すのと並行して、レコードに付加されているアトリビュートによってハッシュをかけ、同じハッシュ値を持つレコードが同じ要素プロセッサに集まるように分配してから（パーティショニング）、つきあわせ（結合演算）をする。このとき大量のデータがネットワークを流れるので、いかにしてトラフィックを減らすかが焦点となり、それはパーティショニングにかかる時間を左右する。従って、シミュレーションではパーティショニングが始まってから終るまでの仮想時間を測定した。なお、もとのGRACE hashでは概念的なデータの供給源はひとつであるが、ハイパーキューブに適用するにあたって、ディスクは各要素プロセッサに付いているものとした。

[1]の最適配置は複アトリビュートのレコードをパーティショニング時に転送距離が小さくなるように、あらかじめレコードを最適に配置しておくものである。

N次元のキューブにおけるアルゴリズムは以下の通り。ひとつのタプルAの各アトリビュートa_iをハッシュし（アクセスされる確率は$p(a_i)$）、ハッシュ値を求める。ハッシュ値をビット列にして$[h_{iN-1},\ldots,h_{ij},\ldots,h_{i0}]$とする。

$$X(j)=\sum_{a_i\in A}\{h_{ij}\times p(a_i)\}$$

とすると、格納ノードのIDのビットj（jは0からN-1）は、

$$bit(j)=\begin{cases}1 & X(j)>0.5\\ 0 & X(j)<0.5\\ 1\text{ or }0 & X(j)=0.5\end{cases}$$

となる。

たとえば、2次元のキューブで3アトリビュート（p = 0.6、0.3、0.1）のレコードを配置するとする。あるレコードを取った時、各アトリビュートのハッシュ値がh=[01]、[11]、[10]だったとすると、格納ノードのIDのビット0は、

$$X(0)=0.4\rightarrow 0$$

ビット1は、

$$X(1)=0.9\rightarrow 1$$

すなわち、そのレコードの格納ノードのIDは[01]となる。

[0] Precise simulation of join on a cube
Effectiveness of optimal placement
S.Hirano, W.Yang, M.Kitsuregawa, M.Takagi
The Institute of Industrial Science, University of Tokyo

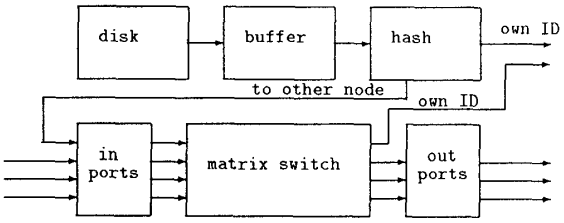

図 1: node structure of 3 dimensional hypercube

3 シミュレーション・モデル

[図1]にシミュレーション・モデルを示す。モデルはC++で記述され、3次元（8ノード）から、7次元（128ノード）まで可変である。ひとつのノードは次の構成要素からなっている。

- ディスク
- ディスク・バッファ
- ハッシュ機構
- 入力ポート
- マトリクス・スイッチ
- 出力ポート

各ノード及びノード構成要素は、グローバルなクロックで同期を取りながら、並列に動作する。ディスクからの入力、ポートの入出力には定められた転送速度があり、1バイトを転送するのに数クロックを要する。その他の構成要素は十分速いと仮定する。

ディスクには結合演算の対象のリレーションが分割されて格納されている。実験の種類により、データには適当なアクセス確率を想定して最適配置をしたものと、ランダムに配置したものがある。ディスクバッファはレコード10個分のバッファを持ち、常にバッファがフルになっているようにレコードを読む。ディスクの転送速度には限りがあるが、シーク時間は0でデータの要求が来るとすぐに転送を開始する。

ハッシュ機構はディスクからレコードを読みながら選択演算によってフィルタを掛け、指定されたアトリビュートをキーとしてハッシュを行う。ハッシュ関数は最適配置の時と同じにする必要があり、今回はノード数の余りである。もし、ハッシュ値が自ノードのIDを示していたらそのレコードは捨て[1]、他ノードならば入力ポートに転送する。入力ポートのキューに空きがなくても、他ノード宛のレコードが発生するまでハッシュを続ける。

入力ポートはキューブの次元数個の隣接ノードとのポートと、ハッシュからのポートを持つ。各ポートにはレコード4個分のキューがついている。

マトリクス・スイッチは入力ポートへ転送されてきたレコードの宛先を見て、他ノード宛ならば中継なので出力ポートを選ぶ。ルートは最短パスの隣接ノードのID）のうち最も小さいものを選ぶ。（デッドロックを防ぐため。）ただし、出力ポートのキューに空きがなかったらそのレコードは入力ポートに留ま

[1] ディスクへのライトは行わない。

る。
出力ポートはキューブの次元数個のポートを持つ。入力ポートと同様に各ポートにはレコード4個分のキューがついている。転送速度は1バイトあたりのクロック数としてパラメータで与え、ディスクの転送速度との相関関係を持たせる。

4 シミュレーション

以下の条件で1リレーションをパーティショニングをするのに要したクロック数を測定した。[図2、3、4、5]

- レコード長 25 バイト
- アトリビュート数 4 個 (一様乱数)
- レコード数 キューブあたり 2560 個
- 選択率 100 %
- ディスクはネットワークより2倍または10倍速い
- キューブの次元数 3 から 7 次元

レコードの個数をキューブ全体で一定にしているので、次元が高いほど各ノードのディスクに格納されるデータは少なくなる。グラフは、

- 最適配置をしない場合 (図中 no optimize の線)
- 4:3:2:1 の割合で各アトリビュートがアクセスされる場合 (図中 4:3:2:1 の線)
- 8:2:0:0 の割合で各アトリビュートがアクセスされる場合 (図中 8:2:0:0 の線)

を示している。
まず、[図2] は次元を変えた時の平均レコード転送距離 (ノード数) である。最適配置をしない場合は次元数の半分になる。
[図3] はディスクがネットワークより2倍速い場合の、100クロックあたりに処理されたレコード数である。このグラフはパーティショニングにかかった時間を直接反映している。
[図4] は [図3] と同様であるが、ディスクはネットワークより10倍速い。[図3] より効果が著しい。平均転送距離の減少ほどには効果が現れないのは、ネットワーク容量がまだ空いているにもかかわらず [図5: ネットワーク使用率]、単純なルーティング・アルゴリズムを使っていることにより、ポートのキューで直列化が起きるためである。

5 結論

ハイパーキューブのネットワークにおけるジョインでは、ネットワークが遅い場合、アトリビュートの個数が少ない場合、また、アトリビュートのアクセス・パターンに偏りがある場合に、最適配置が大きな効果を持つことがわかった。ルーティング・アルゴリズムを改良し、デッドロックなしで、ネットワークを効率的に使用すれば、さらに効果が上がると考えられる。

参考文献

[1] 楊、喜連川、高木、「ハイパーキューブ結合関係データベースマシンにおけるデータ配置方式」情報処理学会第35回全国大会、1987

[2] M.Kitsuregawa,H.Tanaka,T.Moto-oka, Architecture and Performance of Relational Algebra Machine GRACE, Proc. of Int.Conf. on Parallel Processing,1984

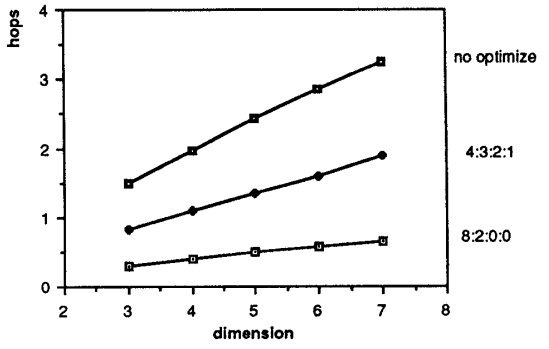

図 2: 平均転送距離

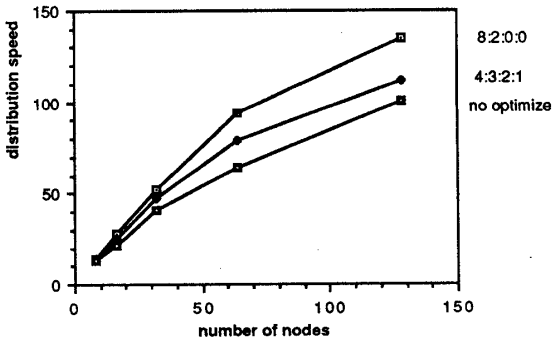

図 3: 100 ステップあたりの処理レコード数 (2 倍)

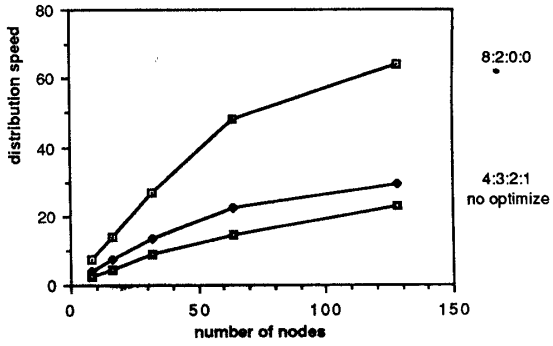

図 4: 100 ステップあたりの処理レコード数 (10 倍)

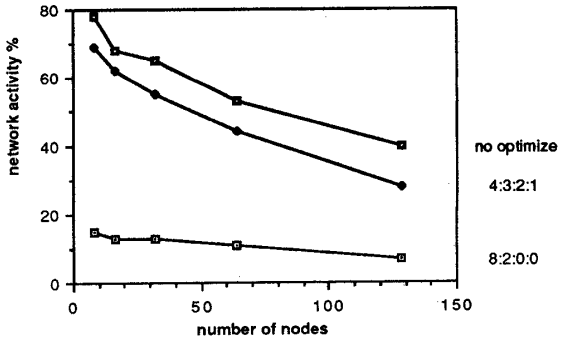

図 5: ネットワーク使用率 (10 倍)