1F－9

関数型並列データベース・システムにおけるメモリ資源割り当て方式

村岡 正則、小川 直子、佐藤 聡、清木 康
筑波大学 電子・情報工学系

1 はじめに

データベースの多様な応用分野に適応可能な並列処理方式として、ストリーム指向型並列処理方式を提案し、それを実現する関数型並列データベース・システムSMASHを構築している[1]。本システムの特徴は、関数型計算の概念を適用することにより、任意のデータベース演算を関数として定義し、それらの演算を組み合わせることによって生成される問い合わせを並列に処理する点にある。また、データベース処理におけるデータの流れ、すなわちデータベース本体および問い合わせ処理途中の中間結果をストリームとして扱っている点が特徴である。

本システムでは、引数の評価方法として要求駆動型評価方式を用いている。この方式では、関数の入力引数の値が必要になった時点で、その引数値の計算を行なう関数に対してデータの生成を要求するデマンドを伝達する。そのデマンドを受けとった関数は、ある一定量（グラニュラリティ）のストリーム要素群を生成する。

本システムでは、各問い合わせの処理に適したグラニュラリティを柔軟に設定することが可能であり、この設定に従って関数間でストリーム要素を受け渡すためののバッファにメモリが割り当てられる。ストリーム指向型並列処理では、グラニュラリティの設定、すなわちバッファへのメモリ資源の割り当てが、問い合わせの総計算量および並列性を決定する主要な要素となる。従って、問い合わせの実行に求められる要求に応じて適切なバッファ・サイズを設定することが重要である[2][3]。

本稿では、メモリを共有していないプロセッサ（以下、サイトと呼ぶ）をネットワークによって結合した環境上での問い合わせの処理時間を最短にするためのメモリ資源割り当てについて述べる。

本方式が対象としているデータベース処理は、データベース演算をノードとする木構造として表される。また、データベース演算は、図1に示すように各サイトに割り当てられる。

2 メモリ資源割り当て方式

データベースへの問い合わせ処理を行う場合、問い合わせの処理時間は、その問い合わせの総計算量と並列性の二つの要素から決定される。

問い合わせの処理時間を決定しているバッファ・サイズ以外のパラメータは、問い合わせが決定した時点で決定あるいは推定することが可能である。また、あるサイトにおける各バッファの大きさは、そのサイトが使用するメモリの総量が決定すれば、[3]で提案されている方式を用いて求められる。

A computational method for memory allocation in the parallel database system based on functional computation
Masanori Muraoka, Naoko Ogawa, Akira Sato and Yasushi Kiyoki
Institute of Information Sciences and Electronics,University of Tsukuba, Tsukuba city,Ibaraki 305,Japan

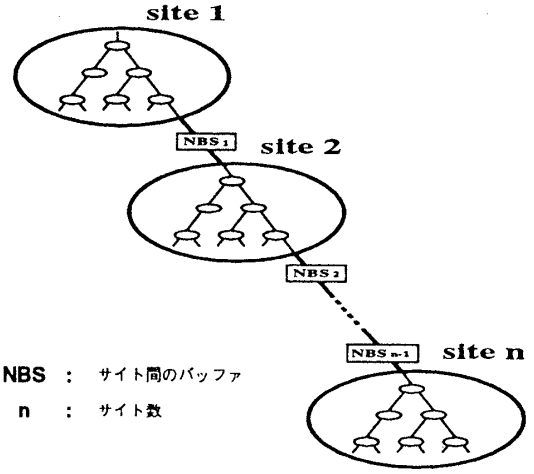

図1: 問い合わせの構造

本稿では、各サイトが使用するメモリの総量を決定する方式について述べる。

2.1 問い合わせの処理時間

問い合わせの処理時間T_{QE}は、ある1サイトにおいて、プロセッサがデータベース演算を実行している時間（実行時間T_{SE}）と実行せずにアイドル状態となる時間（アイドル時間T_{SI}）の和として表すことができる。

$$T_{QE} = T_{SE_i} + T_{SI_i} \ (1 \leq i \leq n)$$

サイトiにおけるアイドル時間T_{SI_i}は、問い合わせ処理の開始時においてNBS_iに最初のストリーム要素群が格納されるまでにアイドル状態になる時間T_{SIb_i}、問い合わせ処理が定常状態にある時にデータ生成のタイミングによってアイドル状態になる時間T_{SIx_i}、サイトiに割り当てられている演算が終了してから問い合わせ処理全体が終了するまでの時間T_{SIe_i}の3要素に分けることができる。

$$T_{SI_i} = T_{SIb_i} + T_{SIx_i} + T_{SIe_i} \ (1 \leq i \leq n)$$

T_{SIb_i}とT_{SIe_i}は、問い合わせの形とバッファ・サイズに依存するものなので、これらを求めるための一般的な式を導き出すことは容易である。しかし、T_{SIx_i}は、問い合わせの中間結果が生成されるタイミングや各サイトのスケジューリングの方法などに依存しているため、その時間を見積もるための一般的な式を導き出すことは難しい。

本方式が対象とする問い合わせ処理では、ボトルネックとなるサイトの影響が他のサイトの定常状態におけるアイドル時間を決定している主な要素となる。従って、ボトルネック・サイトには定常状態におけるアイドル時間は発生しないと仮定すると、サイトiの定常状態におけるアイドル時間T_{SIx_i}は以下のように近似できる。

$$T_{SIx_i} \approx \max_{1 \leq k \leq n} \{T_{SE_k} + T_{SIb_k} + T_{SIe_k}\} - (T_{SE_i} + T_{SIb_i} + T_{SIe_i})$$

これより、サイトiにおけるT_{SE_i}、T_{SIb_i}、T_{SIe_i}の合計をT_{QE_i}とおくと、T_{QE}は、$T_{QE_i}(1 \leq i \leq n)$の最大値となる。

$$\begin{aligned} T_{QE_i} &= T_{SE_i} + T_{SIb_i} + T_{SIe_i} \\ T_{QE} &= \max_{1 \leq i \leq n} T_{QE_i} \\ &= \max_{1 \leq i \leq n} \{T_{SE_i} + T_{SIb_i} + T_{SIe_i}\} \end{aligned}$$

これらにより、ボトルネック・サイトにおける問い合わせの処理時間を最小にするようなメモリ資源割り当てが、最適なメモリ資源割り当てとなる。

2.2 アルゴリズム

問い合わせの処理時間を表す式において時間を表している変数$T_{\cdots}$を、メモリ・サイズ変数$x_{\cdots}$を引数とする関数として設定する。

$$T_{QE}(x_1, \cdots, x_n) = \max_{1 \leq i \leq n} \{T_{SE_i}(x_i) + T_{SIb_i}(x_i, \cdots, x_n) + T_{SIe_i}(x_1, \cdots, x_{i-1})\}$$

問い合わせ処理の終了時における各々のサイトのアイドル時間T_{SIe}は、複数のメモリ・サイズ変数を引数とする関数として表されているが、この関数は各サイトに依存して発生するアイドル時間T_{Ie_i}の和として表すことができる。従って、問い合わせの処理時間は以下のように表される。

$$\max_{1 \leq i \leq n} T_{QE_i}(x_1, \cdots, x_n) = \max_{1 \leq i \leq n} \left\{ T_{SIb_i}(x_1, \cdots, x_n) + T_{SE_i}(x_i) + \sum_{j=1}^{i-1} T_{Ie_j}(x_j) \right\}$$

この式において、問い合わせ処理の終了時におけるアイドル時間T_{Ie}に注目してみると、$\sum_{j=1}^{i-1} T_{Ie_j}(x_j)$が$T_{QE_1}, \cdots, T_{QE_n}$に含まれていることがわかる。また、$\sum_{j=1}^{i-1} T_{Ie_j}(x_j)$は、各式の他の部分の値に依存していない。従って$T_{QE}$は、以下のように表すことができる。

$$\begin{aligned} T_{QE} = \max_{1 \leq i \leq n} T_{QE_i}(x_1, \cdots, x_n) = \\ \max\{T_{SE_1}(x_1) + T_{SIb_1}(x_1, \cdots, x_n)\ ,\ T_{Ie_1}(x_1) + \\ \vdots \\ \max\{T_{SE_i}(x_i) + T_{SIb_i}(x_i, \cdots, x_n)\ ,\ T_{Ie_i}(x_i) + \\ \vdots \\ \max\{T_{SE_{n-1}}(x_{n-1}) + T_{SIb_{n-1}}(x_{n-1}, x_n)\ ,\ T_{Ie_{n-1}}(x_{n-1}) + \\ T_{SE_n}(x_n)\ \}\} \cdots \} \end{aligned}$$

本アルゴリズムでは、サイトiに割り当てるメモリ・サイズの候補値について計算を行なう時に、サイト$i+1 \sim n$の値の最大値が最小となるように、サイト$i+1$に割り当てるメモリ・サイズx_{i+1}を決定する。これを、$n, n-1, \cdots, 1$の順に求めていく。このとき、問い合わせ処理の開始時におけるアイドル時間は、主に隣接するサイトから影響されるものであるとする。よって、T_{QE_i}の計算を行なう場合のこの式に現れる$(x_{i+2}, \cdots, x_n)$は、定数として扱われる。

3 割り当て結果

図2に示す問い合わせについて、本方式を適用した場合（A）のバッファ・サイズと処理時間の予測値、割り当ての計算に要した時間を表1に示す。また、比較対象として、サイト間の並列性を考慮していない[3]で提案されている方式を用いた場合（B）と割り当て可能なメモリを全て割り当てた場合（C）の結果を示す。

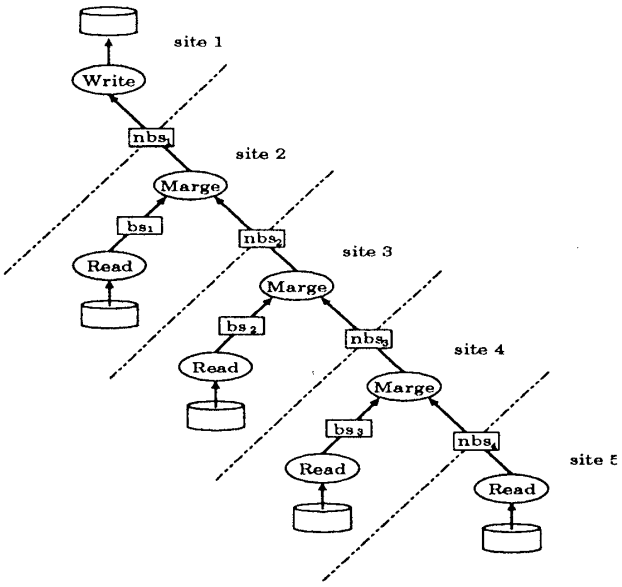

図2: 問い合わせの構成

この予測結果より、提案方式を用いることによって、問い合わせの処理時間を短くすることができ、さらに、メモリの使用量を最小限に押えることができることがわかる。

表1: 割り当て結果（単位: [tuple]）

	site 1	site 2	site 3	予測処理時間	割り当て時間
	$bs1$	$bs2$	$bs3$	[sec]	[sec]
A	384	576	2048	114.9	1.09
B	2752	2752	2752	143.1	0.52
C	4000	4000	4000	159.0	0.00

4 おわりに

本稿では、関数型並列データベース・システムSMASHの問い合わせ処理におけるメモリ資源割り当てを求める方式について述べた。また、本方式を用いた場合の処理時間の予測値を求め、本アルゴリズムの正当性を示した。このアルゴリズムによって、任意のデータベース演算からなる問い合わせについて、問い合わせの処理時間を予測する式から、各バッファへのメモリ資源割り当てを効率良く求めることができる。

参考文献

[1] Y. Kiyoki, T. Kurosawa, K. Kato and T. Masuda. The software architecture of a parallel processing system for advanced database applications. In *Proceedings of 7th IEEE International Conference on Data Engineering*, April 1991.

[2] P. Liu, Y. Kiyoki and T. Masuda. Efficient Algorithms for Resource Allocation in Distributed and Parallel Query Processing Environments. In *Proceedings of the IEEE International Conference on Distributed Computing System*, pp. 316-323, June 1989.

[3] 大西 元, 清木 康. 関数型並列データベース・システムSMASHにおける資源割り当て方式の拡張. アドバンストデータベースシステムシンポジウム論文集, pp. 43-51. 情報処理学会, December 1990.