3Q-9

データベースの多様な応用分野を対象とする並列処理システムSMASH
－メモリ資源割り当ての計算方式の実現－

大西元*，劉澎*，新城靖*，清木康*，益田隆司**．

* 筑波大学　電子・情報工学系
** 東京大学　理学部　情報科学科

1．はじめに

我々は，データベースの応用分野の多様化に柔軟に適応できる並列処理システムの実現を行っている[1]～[4]．すでに，データベースを対象とした任意の基本演算群に内在する並列性を抽出する並列処理方式として，関数型計算モデルにおける要求駆動型評価とストリームの概念を用いたストリーム指向型並列処理方式を提案し，その方式を並列処理環境において実現するための基本プリミティブを提案している[1]．関数型計算モデルは，任意の操作に内在する並列性を抽出するのに適した計算モデルである．そこで，本方式では，関数型計算モデルを並列処理環境を実現するための基本的な計算モデルとして適用している．[1]

ストリーム指向型並列処理方式を実現する基本プリミティブの実現環境として，現在，次の３種類の環境を対象としている．

1）複数台の汎用プロセッサが高速のローカル・エリア・ネットワークに結合された環境．
2）共有メモリをもつバス結合型並列処理マシンの環境．
3）1）および2）を統合した環境．

我々はすでに1）および2）の環境の上で基本プリミティブを実現している．本稿では，1）における各サイト内のメモリ資源，あるいは，2）におけるメモリ資源の割り当てのための計算方式[4]の実現について述べる．

2．ストリーム指向型並列処理方式

ストリーム指向型並列処理方式[1]では，データベースの基本演算は関数として定義され，要求駆動型評価により，関数間において次の２種類の並列性が抽出される．

1）関数の各引数を同時に評価することにより引き出される並列性．
2）関数引数の適用側（消費者関数）とその引数の生成側（生産者関数）との間で抽出されるストリーム型並列性．この並列性は，独立に，生産者関数が一定量（１グラニュラリティ）のストリーム要素を生成し，消費者関数が生成されたストリーム要素に対して操作を行うことによって抽出される．

要求駆動型評価において，関数本体の中で同じ引数を複数回参照しなければならない場合，次の２通りの評価方法が存在する[1]．

1）call-by-name（再計算方式）
2）call-by-need（キャッシング（caching）方式）

本処理方式では，両評価方式を支援しており[1]，ストリーム型の各引数に対して，いずれか一方の評価方法が指定される．本稿では再計算方式を対象にしたメモリ資源割り当ての計算方法の実現方法を示す．

ストリーム指向型並列処理方式の処理系は，次の３要素により実現される[3]．

1）カーネル

関数インスタンスとチャネルの生成および消去，プロセッサのスケジュール，関数インスタンス間の通信の実現等の機能をもつ．

2）関数インスタンス

データベースを対象とする各基本演算は，ストリームを引数とする関数として定義され，関数を単位として基本プリミティブを含む逐次的なオブジェクト・コードへ変換される．それらの基本演算は，並列処理の単位となる関数インスタンスとしてプロセッサへ割り当てられる．

3）チャネル

チャネルを介して，関数インスタンス間でデマンド，ストリーム要素の受渡しが行なわれる．

3．メモリ資源の割り当て計算方式の実現

ストリーム指向型並列処理方式における最適なバッファ資源割り当てのための計算方式を文献[2][4]に示している．ここでは，文献[4]に示した方式によって，図１のようなチェーン型問い合わせの各関数インスタンスへのバッファ資源割り当て量を決定する方法について述べる．

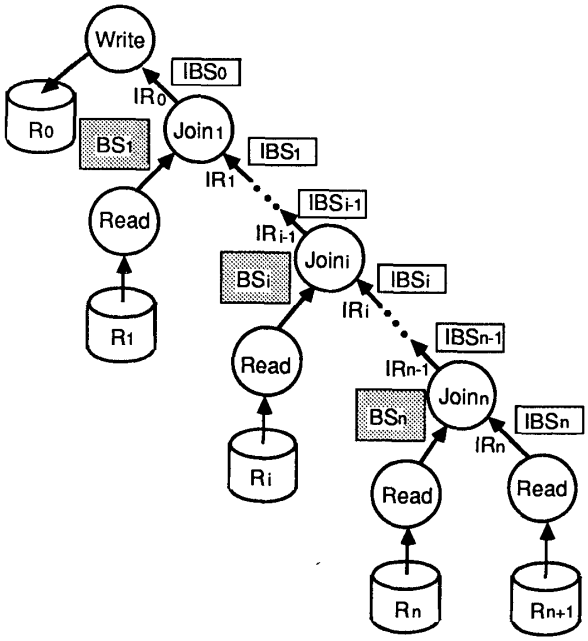

図１　チェーン型問い合せ

3．1　パラメタの設定

問い合わせの実行時間を予測する式に用いられるパラメタについて説明する．（タプル・サイズは固定長とし，また，演算対象属性値のサイズおよび型は同一とする．）

(i=1～n，j=0～n．リレーションとバッファのサイズの単位：タプル)
jsf_i　：結合演算選択率（join selectivity factor）
R_i　　：アウタ・リレーション・サイズ

A parallel processing system SMASH for a wide variety of database applications
--Implementation of the computation method for memory resource allocation--
Hajime OHNISHI*, Peng LIU*, Yasushi SHINJO*, Yasushi KIYOKI*, Takashi MASUDA**
* University of Tsukuba,　** University of Tokyo

R_{n+1} ：最下位の結合演算のインナ・リレーション・サイズ
IR_j ：インナ・リレーション・サイズ
BS_i ：アウタ・リレーションの1グラニュラリティを格納するためのバッファ・サイズ
IBS_j ：インナ・リレーションの1グラニュラリティを格納するためのバッファ・サイズ
c_i ：物理的な制約から決まるBS_iの最小値
iof ：1タプル当りのディスク入出力の平均時間
gpf ：関数インスタンス間の1ストリーム要素（タプルに対応）の受け渡しの平均時間
bsf ：バイナリ・サーチの1回の比較に要する平均時間
qsf ：クイック・ソートを行うために，1ストリーム要素に関して要する平均時間

3．2 各関数インスタンスの実行時間

チェーン型問い合わせにおける各関数インスタンスの実行時間は次の式のように予測される．

1）入出力関数インスタンス

$$f_{read}(r) = f_{write}(r) = r \times iof + r \times gpf$$

2）結合関数インスタンス

$$f_{join}(r, ir, bs, jsf) = (\lceil r/bs \rceil - 1) \times f_{join1}(bs, ir, jsf) + 1 \times f_{join1}(rem, ir, jsf)$$

Where $rem = r - (\lceil r/bs \rceil - 1) \times bs$,

$$f_{join1}(out, in, jsf) = f_{get}(out) + f_{get}(in) + f_{qsort}(out) + f_{bsearch}(out, in, jsf) + f_{put}(out \times jsf \times in),$$

$$f_{qsort}(r) = qsf \times r \times \log_2(r),$$

$$f_{bsearch}(out, in, jsf) = bsf \times in \times (\log_2(out) + out \times jsf)$$

$$f_{get}(r) = f_{put}(r) = r \times gpf$$

3．3 問い合わせの実行時間

図1のようなチェーン型問い合わせにおいて，i番目以下の結合演算を計算する時間は，アウタ・リレーションを入力する時間，および，$Join_i$それ自身の実行時間，インナ・リレーションを$\lceil R_i/BS_i \rceil$回計算する時間の和になる．よって実行時間TCは，以下のような漸化式で表すことができる．

$$TC = f_0 = a_0 + f_1$$

$$f_i = a_i(BS_i) + \lceil R_i/BS_i \rceil \times f_{i+1} \quad (i=1 \sim n)$$

$$f_{n+1} = a_{n+1}$$

Where $a_0 = f_{write}(IR_0)$,

$$a_i(BS_i) = f_{join}(R_i, IR_i, BS_i, jsf_i) + f_{read}(R_i) \quad (i=1 \sim n),$$

$$a_{n+1} = f_{read}(R_{n+1}),$$

$$IR_{i-1} = R_i \times jsf_i \times IR_i \quad (i=1 \sim n),$$

$$IR_n = R_{n+1}$$

ここで，f_iは関数インスタンス$Join_i$の1回の実行に必要な時間を表している．各a_i (i=1～n) はBS_iのみに依存し，他のBS_j (j=1～i-1,i+1～n) には依存しないという特徴がある．

3．4 バッファ割り当て計算のアルゴリズム

ここでの最適なバッファ資源割り当ての計算方式は，アウタ・リレーション・バッファ用の総メモリ資源量BSが与えられた時，

$$BS = BS_1 + BS_2 + \cdots + BS_n$$

の制約条件のもとで，TCを最小とするメモリ資源の割り当て（$BS_1, BS_2, \cdots, BS_n$）を求める方式である．インナ・リレーションバッファ・サイズ（IBS_i）は計算回数には関係しないので計算式の中には現れない．IBS_iは物理的制約によって，あらかじめ設定され，すでに割り当てられているものとする[4]．

以下に，この計算方式の実現の概要を述べる．

3．3で示した式より，f_iは，f_{i+1}とBS_iにより決定されるので，次のような形で表すことができる．

$$f_i = f_i(f_{i+1}, BS_i) \quad (i=1 \sim n)$$

ここで，部分関数f_iの最小値F_iは，uタプル（$1 \leq u \leq BS$）のバッファ資源を利用する時，$Join_i$自身でtタプル使用し，残りのu－tタプルをそれより下位の結合演算（$Join_{i+1} \sim Join_n$）に分け与えるとすると，f_{i+1}の最小値であるF_{i+1}と補助変数tの関数の形で表現することができる．

$$F_i(u) = min\{f_i(F_{i+1}(u-t), t)\} \quad (c_i \leq t \leq BS\text{-}1, i=1 \sim n)$$

$$F_{n+1} = f_{n+1} = a_{n+1}$$

（$F_1(BS)$は問い合わせの実行時間の最小値を表すことになる．）

この式を以下の手順で計算する．

0）F_{n+1}は問い合わせから一意に決まる．

1）最下位の結合演算のF_nから順に，F_1まで求めていく．F_iの計算においてはF_{i+1}の値を用いる．

$$F_i(u) = min\{f_i(u)\} = min\{a_i(t) + \lceil R_i/t \rceil \times F_{i+1}(u-t)\} \quad (u \leq BS, c_i \leq t \leq u-1)$$

ここで，tの候補値となるのは，$\lceil R_i/cc \rceil$である[4]．（cc（再計算回数）は，$\lceil R_i/BS \rceil$以上$\lceil R_i/c_i \rceil$以下の整数）

この時，各uについて，u，$F_i(u)$，tを記録しておく．

2）最後に$F_1(BS)$を計算し，そこで利用する各ステップにおけるtの値が求めるバッファ割り当て（$BS_1, BS_2, \cdots, BS_n$）となる．

4．おわりに

本稿では，データベースを対象とした並列処理システムにおける最適なバッファ資源割り当て方式の実現について述べた．

ここでは，チェーン型問い合わせにおける方式を示したが，任意の構造をもった問い合わせについてもバッファ資源割り当て方式の計算を適用できる[4]．今後は，任意の問い合わせに対応できるように，実現を進めていく．

参考文献

[1]Y.Kiyoki,K.Kato and T.Masuda："A Relational Database Machine Based on Functional Programming Concepts",Proc. ACM-IEEE Computer Society Fall Joint Computer Conf.,pp.969-978

[2]清木，劉，益田：関係演算のストリーム指向型並列処理における資源割り当て方式，情報処理学会論文誌，Vol.28,No.11,pp.1177-1192(1987)

[3]新城，清木，劉，益田：データベースおよび知識ベースを対象としたストリーム指向型並列処理系の共有メモリ・マシン上への実現，情報処理学会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，Vol.88,No.9,Dec 8 (1988)

[4]劉，清木，益田：ストリーム指向型関係演算処理におけるバッファ資源割り当ての計算方式，情報処理学会論文誌，Vol.29,No.8,pp.770-781(1988)