4Q-5

# 並列論理型プログラムにおける新プラグマ方式とその応用

神田　陽治

富士通　国際情報社会科学研究所

与えられた並列論理型プログラムに対して、それをどのように並列マシン上に分散し実行させるかは難しい問題である。本論文では、新しい動的プラグマ方式を提案した。各ゴールは、分散制御リストを親ゴールから受け継ぎ、順次消費していく。途中での制御リストの変更は自由だが、変更の方法に制限を設けることで、大まかな分散の傾向を外部から決定できる特徴を持つ。変更方法を指定するのがプラグマの役目である。

## 1．Shapiro のプラグマ方式〔Shapiro 84〕

p：－q@forward, r.

は、ゴールpが、qとrを生成する点においては、

p：－q, r.

と全く同じである。異なるのは前者ではp、q、rのゴール達を、どのプロセッサに割り付けるかを陽に指定する、プラグマの仕組みが使われている点にある。Shapiro の方法では、すべてのゴールには、

（プロセッサ座標、現在の方向）

なるゴール属性が与えられる。pがqとrを生成する際には、qとrの実行属性はpの実行属性を元に計算される。プラグマは、実行属性計算の方法を指定する。例を挙げるならば、forward は、現在の方向へプロセッサ座標を一つ進めることを、right（left）はプロセッサ座標は同じで、現在の方向を右に（左に）90 度回すことを指示する。

我々は、Shapiro のプラグマを評価する試みの中から、次のような問題点を見つけ出した。

①最適な性能を得るようなプラグマ付けは、場合場合によって異なり、特定できない〔大原８７〕。

②静的なプラグマでは、マルチタスクの環境で負荷を平衡させる（タスクミックス）のは難しい〔神田８６〕。

③ゴールの位置指定ばかりでなく、変数セルの位置指定、また、ゴールの分散タイミング指定（いつゴールを送り出すか）まで含めるべきである〔神田８６〕。

並列ハードウェアを作る立場からは、通信のコストは無視できず、例えばプロセッサ当たり１００リダクションにつき１つのプロセッサ間通信という要求が出てくる。しかし、この種の要求を充たすようにプラグマを付けることは難しい。分散の度合が低すぎるか高すぎるかで、分散しても性能向上に結びつかない場合が多い（①）。

静的なプラグマがコントロールできるのは、一つのプログラムの実行の範囲に限られる。多数のプログラムが同時に実行されるマルチタスクの環境では、資源を有効に使うためにも、オペレーティングシステムが外からプログラムの実行をコントロールできなくてはならない（②）。

またゴールを、照合するヘッドを持つクローズのボディ部で置き換える際、一斉に子ゴールを放ったり、変数セルを実行中のプロセッサに一律に割り付けるのではなく、子ゴールをタイミングを見て放出したり、変数セルを一つ一つ別のプロセッサに割り付けられる自由度が欲しい。通信コストを減らしたいからである（③）。

## 2．新プラグマ方式

静的なプラグマ方式では、プログラムの負荷分散の責任を全てプログラマに任す。しかし細々したところまで面倒みるのでは大変である。良いプラグマ付けが場合場合で違うのなら（①）、是非とも分散方式を固定したい個所のみプログラマがプラグマ指定すれば良いこととし、他は外から分散を指示できるようにする。外からの分散指定はタスクミックスの制御に好都合である（②）。一方、クローズの展開法に関する細かい指定は、与えられたプログラムの範囲で性能が向上するとわかっている分散方式を、細かく指定可能にするために、望ましい（③）。

Shapiro 方式など静的なプラグマ方式に代わる、新しいプラグマ方式を提案する。（次頁の図参照のこと。）

### A．道の教え

Shapiro の静的プラグマ方式の場合は、各ゴールに、

（プロセッサ座標、現在の方向）

なるプロセス属性を割り当てた。新プラグマ方式では拡張して、

（従うべき道、プロセッサ座標、現在の方向）

を使う。ただし、従うべき道には、Shapiro 流の静的プラグマのリストが与えられるとする。例えば、次のPATHは、ジグザグに無限に伸びる道である。

PATH = [right, forward, left, forward, ...]

ここで「... 」は、頭部の繰り返しを表すものとする。

### B．道を究める

新プラグマ方式においても、ゴールと照合するクローズが見つかり、ゴールが子ゴールに展開されるとき、子ゴールたちの制御情報は、元のゴールの制御情報からプラグマが指定した方法で計算される。新プラグマは、p：指示の形で指定する。

a）道を伝授する

p：－q, r.

基本的には、プロセス属性は、親ゴールから子ゴールにそのまま伝えられる。すなわち、並列に実行される子ゴールq, rには、同じプロセス属性が渡される。ただし、

p：－q＆r.

のように逐次実行の指定のときは、ゴールrへ伝えられるのは、ゴールq実行終了後のプロセス属性とする。

b）道に従う

子ゴールには、「：指示」の形のプラグマ指定を、０個以上後置できる。

---

New Pragma Scheme for Parallel Logic Programming and its Application to OS

Youji Kohda

IIAS-SIS, FUJITSU LIMITED

p :－q ：指示, r.

プラグマは、プロセス属性の計算方法を指定するが、子ゴールに「則の道」を歩ませるため、次の三通りの扱い方に制限をする。この制限が新プラグマ方式の核心である。

(1)道をたどる： 例 q ：advance(1)
(2)道を拓く ： 例 q ：bypath(rhs,1)
(3)道草を喰う： 例 q ：square(top-right,1)

道に素直に従い、決められた教えの通りに道をたどるのが(1)の場合である。例えばadvance(1)は、親ゴールから道に沿って 1ステップ進んだところに子ゴールを割り付ける指示である。例えば、先のPATHが親から与えられたとして、PATHより 1ステップ分を切り出し、

PATH = [right,forward ¦ PATH2 ]

子ゴールのプロセッサ座標と現在の方向が更新される。

Goal@(right,forward)

そして、残りの道PATH2 が子ゴールに渡される。

親の道を尊重するものの、子供なりの道に歩み出すことを許すのが(2)の場合である。といっても親の道から大きく逸れることは許さず、並行に走る側道を設けることのみ許す。例えばbypath(rhs,1) は、親の道の右側 1つ離れた道を子供に与える。これは道は親のものと同じで、次のようにしてプロセッサ座標のみ変えることで達成できる。

Goal@(right,forward,left)

親の道を無視し、子供に道を直接与えるのが(3)の場合である。といっても与えるのは、辺りをくるくる回るだけの閉じた道であり、道草を喰うだけのことしか許さない。例えばsquare(top-right,1) は、右上の大きさ 1 の四角形をくるくる回る道を、子ゴールに与える。すなわち、

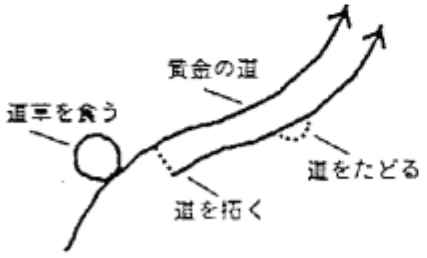
PATH = [forward,right,... ]

を新しい道として、子ゴールへ与える。

C. 黄金の道

オペレーティングシステムに初期ゴールが投入され、プログラムの実行が開始されるとき、オペレーティングシステムは初期ゴールに一つの道を教える。初期ゴールはつぎつぎと新しいゴールに展開されていくが、子ゴールが持たされる道は、すべて（プログラムに付けられたプラグマによって変更を受け、多少の軌道修正はあるせよ、）この道に従う。すなわち、誰もが従うべき至上の道という意味で、この道を黄金の道 (Golden Path)と命名する。

プラグマは道を修正するものの、それからの大きな逸脱は許さない。投入されたゴールから派生するゴール全体をタスクと呼ぶことにすると、タスクは黄金の道を中心とした、大きな流れ(tide)を形成することになる。この特質をマルチタスクオペレーティングシステムに役立てることができる。すなわちタスクを起動しようとするとき、現在のプロセッサ割り当ての状態（タスクミックス）から、空いていそうなプロセッサを数多く含むような軌道を求め、タスクへ黄金の道として教える。これにより大まかにだが、負荷分散が図れる。黄金の道の指定は、（裸の物理マシンからは離れた）論理マシンの形状を外から与えているのだとも言える。




D. クローズの展開法を細部まで指示する

言語仕様を決めるまでにはいたっていないので、使えそうなアイデアを示唆するにとどめておく。

a）変数セルの位置指定

展開中のクローズに属す変数セルを、展開を担当しているプロセッサに割り付ける必然性はない。後で通信が少なくなるようなプロセッサに割り付けるべきである。次のようにすれば、ゴールの位置指定法を流用できる。

変数セルを陽に割り付ける述語assignを用意し、

assign(X):指示

とする。すなわち、まずプラグマによってassign述語自体を目的のプロセッサに送り、そこでassignの実行により変数セルを取る。

b）ゴールの分散タイミング指定

変数の値、問い合わせの通信を減らすためには、ゴールの引数の変数が確定するまで、ゴールの分散を遅らせた方がよい。これはバインドフックの機構で実装できよう。例えば、

wait(X,Goal:指示)

変数Xが確定した後、Goalを指示により分散配置する。

3. 例題

行列の掛算を行うConcurrent Prolog プログラムである。mmはbypath(rhs,1) を繰り返して、行列の行数だけの幅を持つ道を作りだす。vmはadvance(1)を繰り返して、道を1ステップずつ進み、進むたびにipを置いていく。ipはその場で行列要素を計算する。（下図参照のこと。）

```
mm( [] ,Ym, [] ).
mm( [Xv ¦ Xm] ,Ym, '[Zv ¦ Zm] ):-
    vm( Xv,Xm?,Zv),mm(Xm?,Ym,Zm):bypath(rhs,1).
vm( Xv, [] , [] ).
vm( Xv, [Yv ¦ Ym] , [Z ¦ Zv] ):-
    ip( Xv?,Yv?,Z),vm(Xv,Ym?,Zv):advance(1).
ip(Xv,Yv,Z):-
    ip1(Xv,Yv,0,Z).
ip1( [X ¦ Xv] , [Y ¦ Yv] ,Z0,Z):-
    Z1 is (X*Y)+Z0, ip1(Xv?,Yv?,Z1,Z).
ip1( [] , [] ,Z,Z).
```

本研究は、第５世代コンピュータ・プロジェクトの一環として実施しています。

〔Shapiro 84〕 Systolic Programming, in FGCS,1984.
〔大原87〕 核言語ＫＬ１分散方式 本論文集6V-2.
〔神田86〕 並列論理型プログラミング言語における
プラグマ機能について、
日本ソフトウェア科学会第３回大会,1986.