2Q-6

並列処理記述言語PARAGRAMにおけるプロセッサ割付方式

山本富士男　㈱日立製作所・中央研究所

1．はじめに

PARAGRAMは、記述性の向上と演算並列化の促進を目的とする数値シミュレーション向き高水準並列処理記述言語であり、そのトランスレータにおけるプロセッサ割り付け方式の概要についてはすでに報告した。[1] 今回は、この割り付けアルゴリズムの数学的根拠と他のより簡単な割り付け方式との比較評価を中心に述べる。

2．プロセッサ割付アルゴリズムSDAの概要

PARAGRAM言語では、多様な並列構造を許容するため、並列ループpcalcfor（同種手続きの並列実行）と並列ケースpcalc case（異種手続きの並列実行）および逐次ループを混在して使えるようにしている。SDA（Steepest Descent Based Allocation）は、メモリ共有型マルチプロセッサを想定して、これらの記述が任意にネストする並列構造に対するプロセッサ割り付けを行なう。

その概要を例題によって説明する。図１に多重並列構造の記述例とその構造グラフを示した。｛｝の中の数値はループボディの演算量を示している。delay(a)は、並列ループ pcalc forの繰り返しの間で規則的な依存関係があって、実行開始可能時間がａづつ遅れること（いわゆるdoacrossループ）を意味する。構造グラフのノードは、PF(pcalcfor)、PC(pcalccaseのルート）、PCB(pcalccaseの分岐)、SL（逐次ループiter）の４種類である。このプログラムを４台のプロセッサで実行させる場合を考える。

まず、最深レベルのノード（level k=4）のループx,y各々について、プロセッサ台数を1～4まで変化させた場合の並列実行時間を算出して図２-step1のようなcostの表を作成する。次にlevel k=3、さらにk=2のノードへ進む。この時の最短時間計算は、Polychronopoulos[2]に基づいて行なう。

このあと、レベル k=1のpcalc caseルートノードとしての最短並列実行時間を求めるのがこのSDA の本質的な部分である。まず初期割り付けを、pcalc caseの各分岐に１台づつとする。次に、何れかの分岐に１台割り付け台数を増やす場合（３つの候補がある）のうち、最も計算時間削減量が大きくなる場合を見つける。その結果がstep4 ②に示すものである。この割り付け状態に対し、さらに同様に一台割り付け台数を増やす手続きを、計算時間の極小値が得られるまで続行する。この場合は、図２に示すように３回の探索で解が求められる。

3．SDAの数学的根拠

上記の極小値探索法の根拠は、数値解析における最急降下法にある。今、レベルｋにあるpcalc caseの分岐ノードjに割り付けらるプロセッサ台数を$L_{k,j}$とすると、レベルk-1にあるpcalccaseのルートノードの計算時間T_{k-1}は

$$T_{k-1}=f(L_{k,1}, L_{k,2}, \ldots, L_{k,j}, \ldots, L_{k,m})$$

の形に表せる。もし、$L_{k,j}$ の値が連続空間で変動し、この最小値を最急降下法で求めるのならば、勾配ベクトル

$-grad\ f(L)$の成分比に従って$L=(L_{k,1}, L_{k,2}, \ldots, L_{k,m})$を修正すべきである。ここでは、$L_{k,j}$は正の整数であることから、このように修正することを、

$$\Delta T_{k-1}=f(L_{k,1}, L_{k,2}, \ldots, L_{k,j-1}, L_{k,j}, L_{k,j+1}, \ldots, L_{k,m}) - f(L_{k,1}, L_{k,2}, \ldots, L_{k,j-1}, L_{k,j}+1, L_{k,j+1}, \ldots, L_{k,m})$$

をj=1～mに関して最大にすることで近似する。すなわち、何れかの分岐$L_{k,j}$に一台割り付け台数を増やす場合のうちΔT_{k-1}が最大となるものを採用するわけである。以上のSDA法の実現に必要な、“$L_{k,j}$に一台増やした場合の割り付け”は CPMVP（Critical Path Method with Virtual Prosessors）と呼ぶ方式で決定する。詳細は略すが、これは通常のCritical Path法と異なり、case(i)に複数台が割り当てられる場合は、その台数分が空くまで待たせるものとする。また、優先度の低いcase分岐（この場合はcase(1)）も、優先度の高い他のcase 分岐の実行開始が遅れない範囲で先に実行するよう割り当てる。

SDA 方式の演算量は、並列ループ、逐次ループに関しては、その総数をｒ、プロセッサ台数をｐとした場合、$O(r\cdot P^2/2)$である。[2] また、case ルートノードについては、その分岐個数をｃとした場合、図２の手続きから、最悪の時$O(P\cdot(P+1)\cdot C^2/2)$回のCPMVPの適用が必要となる。しかし、この回数は極小値への収束性で変わる。また、SDA において、極小値に収束した後も可能な限り計算を続行すると次節で述べる簡易割付法IAAと同じ結果に到達する。したがって、IAAは本SDAの割り付けで決まる実行性能の一つの下限を与えると言える。

4．簡易割り付け方式との比較

本方式を、別の単純な割り付け方法IAA（Iterative Allocation of All Procssors）と比較した結果を図３に示した。IAAは各case 分岐に常に全プロセッサを割り付け、そ

A Processor Allocation Algorithm in PARAGRAM Language Translator

Fujio YAMAMOTO

Central Research Laboratory, HITACHI Ltd.

れを順次実行するものである。プロセッサ台数の増大と共にSDAによる割り付けの優位性が顕著となることがわかる。

5．おわりに

タスク起動、終了等のオーバヘッド、およびプロセッサエレメントのベクトル加速性能等も、それらの効果をループ本体の演算量に反映させることで取り扱うことができる。SDA は、常に最適割り付けを与える保証はないが、上記のように、比較的少ないコストで良い割り付けが得られると期待できる。実問題での性能評価は今後の課題である。

なお本研究は、通商産業省工業技術院大型プロジェクト「科学技術用高速計算システムの研究開発」の一環として実施されたものである。

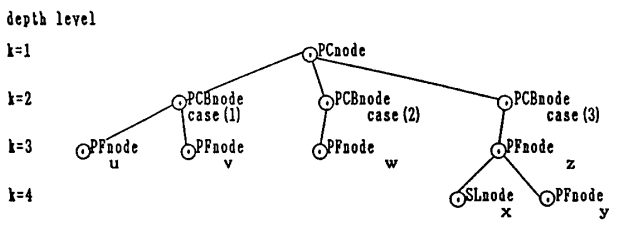
参考文献

1）山本、“多重並列プログラムに対するプロセッサ割り付け方式の検討”、情処学第37回全国大会講演予稿集、pp. 670-671

2）C. D. Polychronopoulos et al. ; "Executon of Parallel Loops on Parallel Processor Systems", Proc. of ICPP' 86, pp. 519-527, 1986

```
PCALC  CASE;
  CASE(1):
    PCALC  FOR  u  IN   [1:450];
      delay(0)
      (5)
    END  PCALC;
    PCALC  FOR  v  IN   [1:45];
      delay(0)
      (55)
    END  PCALC;
  CASE(2):
    PCALC  FOR  w  IN   [1:630];
      delay((w-1)*9)
      (25)
    END  PCALC;
  CASE(3):
    PCALC  FOR  z  IN   [1:7];
      delay(0)
      ITER  FOR  x  IN   [1:50];
        (13)
      END  ITER;
      PCALC  FOR  y  IN   [1:113];
        delay(0)
        (9)
      END  PCALC;
    END  PCALC;
END  PCALC;
```

（a）PARAGRAM言語による並列構造記述例



（b）プログラム構造グラフ

図１　多重並列構造の例

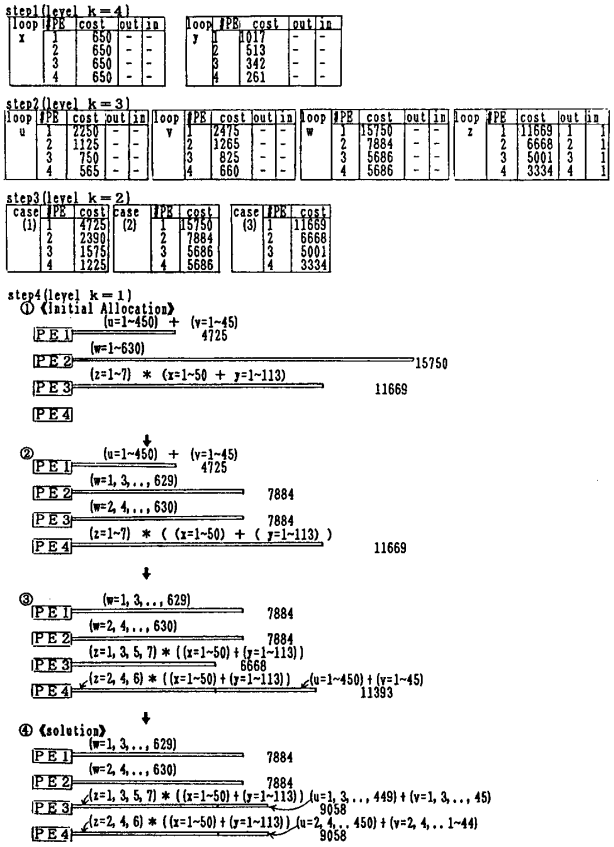
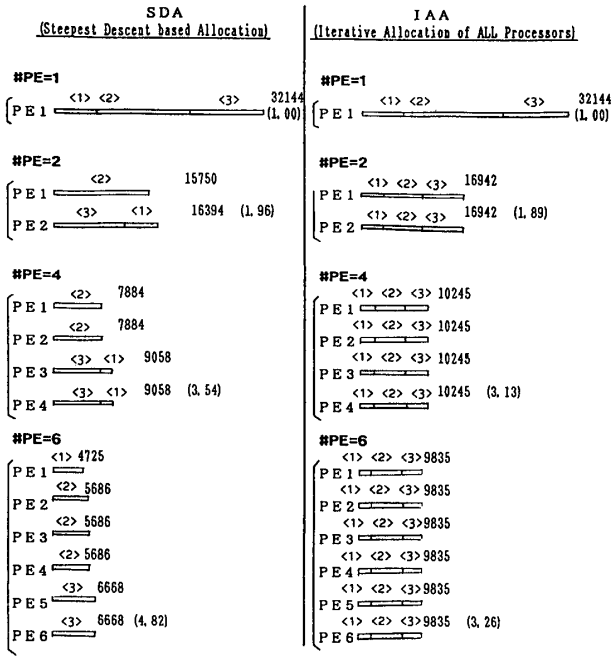


図２　ＳＤＡ法によるプロセッサ割り付け手順



（注）〈 〉内の数値は、図１でのCASE分岐番号を表す。

図３　簡易割り付け法との比較