# 4J-05 マルチグレイン並列処理における階層的並列処理のためのプロセッサクラスタリング決定手法

山本 正行†, 山本 晃正‡, 小幡 元樹†, 笠原 博徳†
† 早稲田大学理工学部電気電子情報工学科, ‡ 松下電器産業 (株)

## 1 はじめに

マルチプロセッサシステムを容易かつ高実行効率で使用するために自動並列化コンパイラの開発が進められている。しかし、現在までに開発されている自動並列化コンパイラのほとんどはループの自動並列化に焦点を当てているが [2, 3]、この技術は成熟期に入っており、今後大幅な性能向上は難しいと言われている。そこで今後の実行性能の向上のためには、プログラム中の基本ブロック、ループ、およびサブルーチン間の粗粒度並列性、ループのイタレーション間の中粒度並列性、ステートメント間の近細粒度並列性に着目し、それらを階層的に組み合わせてプログラム全域の並列性を利用するマルチグレイン並列処理 [1, 4, 7] が研究されている。このマルチグレイン並列処理において効率良い並列処理を実現するためには、各階層（ネストレベル）の並列性に応じた階層的なプロセッサクラスタリングが重要となる。本稿ではプログラム中の各部分階層に適した並列処理手法とそのためのプロセッサクラスタリング自動決定手法を提案する。

## 2 マルチグレイン並列処理

本節では、マルチグレイン並列処理の主要構成要素である粗粒度、中粒度、近細粒度並列処理について説明する。粗粒度並列処理のためのマクロデータフロー処理では、プログラムを次に示す 3 種類のマクロタスク (MT) に分割する。

- BPA (基本ブロック，及び複数の小基本ブロックを融合したブロック)
- RB (最外側ナチュラルループ)
- SB (インライン展開が有効でないサブルーチン)

コンパイラは、MT 間のコントロールフロー解析やデータフロー解析からマクロフローグラフ (MFG) を生成する。次に MFG を元に最早実行可能条件解析が行われ、MT 間の並列性を抽出したマクロタスクグラフ (MTG) を得る。生成された MT は各階層の MTG において実行不確定性 (条件分岐等) が存在する部分にはダイナミックスケジューリングを用い、それ以外の部分にはスケジューリングオーバーヘッドの小さいスタティックスケジューリングを用いて、プロセッサエレメント (PE) をグループ化したプロセッサクラスタ (PC) に割当てられる。さらに MTG 中の各 MT に注目すると、MT が BPA ならば BPA 内部のステートメントレベルの近細粒度並列処理が、RB ならばループ並列処理やシーケンシャルループに対するボディ部の近細粒度並列処理あるいはマクロデータフロー処理、SB もマクロデータフロー処理が適用され、階層的にマクロデータフロー処理が行われる (図 1 )。

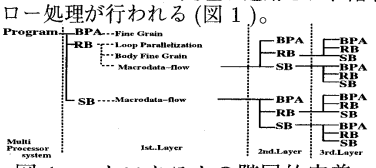


図 1: マクロタスクの階層的定義

また MT の階層的な定義に対応して PC も階層的に定義される。第 0 階層をマルチプロセッサシステム全体と定義し、第 1 階層 PC を、第 0 階層 PC 内のプロセッサエレメント (PE) をグループ化して定義される PC とする。同様に第 M 階層 PC は第 M-1 階層 PC 内の PE をグループ化することで階層的に定義できる。そして各階層の粗粒度並列処理はその階層の PC 間で処理され、MT 内部は PC 内の PE 或いは下位階層で定義される PC 間で並列処理される。

階層的マクロデータフロー処理はプログラムのネストレベルにより複数階層で定義でき、その階層の並列性を最大限に利用可能な並列処理手法の選択と PC の割当て数の決定は実行性能に大きな影響を与える。

---

* A Processor Clustering Scheme for Hierarchical Parallel Processing in Multi-Grain Parallel Processing
Masayuki YAMAMOTO†, Terumasa YAMAMOTO‡, Motoki OBATA†, Hironori KASAHARA†
† Department of Electrical, Electronics and Computer Engineering, Waseda University
‡ Matsushita Electric Industrial Co.,Ltd.

## 3 プロセッサクラスタリング決定手法

本節では、各階層における PC 割当て数の決定手法について説明する。

プログラム全体において効率良い並列処理を実現するための PC 構成（プロセッサクラスタリング）の決定は、各階層毎の並列性を考慮して行われねばならない [10] 。

ここで、$MT^0$ はプログラム全体、$MT_4^1$ は第 1 階層 (メインルーチン) の 4 番目の MT のように粗粒度タスクの階層性を表現することとする。すなわち $MT^k_{i_1,i_2,...,i_k}$ は $MT^{k-1}_{i_1,i_2,...,i_{k-1}}$ の内部で生成されるマクロタスクグラフ (MTG) の $i_k$ 番目の MT を表すものとする。また、$MT^0$ 内部をマクロタスクに分割して生成される MTG を $MTG^1$ と表現し、$MT^k_{i_1,i_2,...,i_k}$ の MTG を $MTG^{k+1}_{i_1,i_2,...,i_k}$ と表す。この表記法を用いると、$MTG^{k+1}_{i_1,i_2,...,i_k}$ の並列性指標 (Para) は次のようになる。

$$\mathrm{Para}^{k+1}_{i_1,i_2,...,i_k} = \frac{MTG^{k+1}_{i_1,i_2,...,i_k} \text{ の 1PC での逐次処理時間}}{MTG^{k+1}_{i_1,i_2,...,i_k} \text{ のクリティカルパス長}}$$

この式は直観的には、$MT^k_{i_1,i_2,...,i_k}$ が単一の並列実行可能ループであればその回転数を、シーケンシャルループあるいはサブルーチンであれば複数の MT からなり内部へマクロデータフロー処理が適用される場合には $\lceil \mathrm{Para}^{k+1}_{i_1,i_2,...,i_k} \rceil$ がクリティカルパス長で並列処理を行うためのプロセッサ数の下限を表しているということが分かる。

しかし、$MTG^{k+1}_{i_1,i_2,...,i_k}$ が複数の MT から成りかつ内部の MT に並列化可能ループが含まれている場合に、効率の良いマルチグレイン並列処理を実現する為には、並列化可能ループの MT 分割により負荷分散を図る必要がある。

しかし、並列化可能ループの分割は使用するマルチプロセッサのローカルメモリ容量、キャッシュ容量を考慮してミドルパスにおけるデータローカライゼーションフェイズで行われる。従ってミドルパスの最初でどの階層の並列性をどのような PC 構成で利用すべきかという本クラスタリングにおいては、その階層での並列性を十分利用し、ローカライズフェイズと矛盾無く適切なクラスタリングができるかを考える必要がある。

### 3.1 クラスタリング決定の手順

プログラム全体で使用可能な PE 数を $PEnum^0$ と表し、$MT^k_{i_1,...,i_k}$ を実行するために使用可能な PE 数は $MT^k_{i_1,...,i_k}$ の属する MTG の PE 数であるので、$PEnum^k_{i_1,...,i_{k-1}}$ と表現することとする。ここで $MT^k_{i_1,...,i_k}$ 内部のマクロタスクグラフ $MTG^{k+1}_{i_1,...,i_k}$ を実行する場合のクラスタリングを考えると、生成すべき PC 数を $PCnum^{k+1}_{i_1,...,i_k}$ とした時 $MTG^{k+1}_{i_1,...,i_k}$ 内の MT を実行する PE 数は $PEnum^{k+1}_{i_1,...,i_k}$ と表現でき、$PEnum^{k+1}_{i_1,...,i_k} = PEnum^k_{i_1,...,i_{k-1}} / PCnum^{k+1}_{i_1,...,i_k}$ である。これを $MTG^{k+1}_{i_1,...,i_k}$ におけるクラスタ $(PCnum^{k+1}_{i_1,...,i_k}, PEnum^{k+1}_{i_1,...,i_k})^{k+1}_{i_1,...,i_k}$ と表記する。

ここで生成すべき $PCnum^{k+1}_{i_1,...,i_k}$ は一意に定まらないため、$Para^{k+1}_{i_1,...,i_k}$ と $PEnum^k_{i_1,...,i_{k-1}}$ を元に、考え得るクラスタ構成におけるコストを算出し最小のものを選択する。この時 クラスタリング決定の具体的な手順は次のようになる。

1. 一般的に外側階層の MT の処理時間は大きいため、粒度の粗い外側階層になるべく多くのプロセッサを割当てた方が相対的にスケジューリングオーバーヘッドを小さくできる。よって粗粒度並列性を最大限に利用可能なクラスタ構成 $(PEnum^k_{i_1,...,i_{k-1}}, 1)$ から順に試行する。
2. 仮想的なクラスタを (vPCnum, vPEnum) と表した時、内部に並列実行可能ループ $MT^{k+1}_{i_1,...,i_k,l}$ ($MTG^{k+1}_{i_1,...,i_k}$ 内の l 番目の MT が並列実行可能ループであるとする) が存在

する場合には、vPCnum と $Para^{k+1}_{i_1,...,i_k}$ より 3.2 で述べる手法で $MT^{k+1}_{i_1,...,i_k,l}$ の仮分割数候補を求め、再度 MTG を生成する。この時の 仮分割数を $vDnum^{k+1}_{i_1,...,i_k,l}$ とする。$MTG^{k+1}_{i_1,...,i_k}$ を (vPCnum, vPEnum) で実行する場合のスケジューリング長を求める。これを $t_{schedule}$ と定める。

3. $t_{schedule}$ を 3.3 で述べる手法で評価し、一定条件を満たさない場合は手順 2 から次のループの分割数候補で繰り返し、さらに条件を満たさない場合には手順 1 から次のクラスタ候補で繰り返す。
4. 手順 1 から 3 を $MTG^{k+1}_{i_1,...,i_k}$ の各 $MT^{k+1}_{i_1,...,i_k,i_{k+1}}$ 内部の $MTG^{k+2}_{i_1,...,i_k,i_{k+1}}$ に階層的に適用する。

以上でプログラム全域でのクラスタリングが決定する。

### 3.2 並列化可能ループの分割数

まず $MTG^{k+1}_{i_1,...,i_k}$ について、提案するクラスタリング決定手法における分割数決定に用いるクリティカルパス (CP) 長 ($t_{cr}$)、並列処理によって得られうる最短実行時間 ($t_{min\text{-}exe}$) 及びそれらの差 ($t_{diff}$) を定義する。

$t_{cr}$ : $MTG^{k+1}_{i_1,...,i_k}$ 内部の $MT^{k+1}_{i_1,...,i_k,j}$ のうち並列実行可能ループのコストを (1PE での実行コスト / $PEnum^{k}_{i_1,...,i_{k-1}}$) とおいて $MTG^{k+1}_{i_1,...,i_k}$ について算出したクリティカルパス長

$$t_{min\text{-}exe}: \frac{MTG^{k+1}_{i_1,...,i_k} \text{中の全 } MT^{k+1}_{i_1,...,i_k,j} \text{の 1PE での実行コストの和}}{PEnum^{k}_{i_1,...,i_{k-1}}}$$

$t_{diff}$ : $t_{min\text{-}exe} - t_{cr}$

クリティカルパス上の並列化可能ループは可能な限り早く終了することが望ましいので、vPCnum 個に分割すべきである。ここで $t_{cr}$ と $t_{min\text{-}exe}$ の差 $t_{diff}$ の正負によりそれぞれの場合に対し 2 つの手法を提案する。

#### 3.2.1 IdlePC 数に基づく分割数の決定

本手法では、$t_{diff} \leq 0$ ,$t_{diff} \simeq 0$ すなわち vPCnum に比べて並列性が比較的小さい場合、CP 上に無い並列化可能ループは CP 上の MT との並列性を活かすために vPCnum−1 個に分割してコストを計算することが考えられる。そこで最初に vPCnum−1 分割を 1 番目の分割候補とする。

次候補は分割前の並列性を活かす立場で、さらに 1 つ少ない分割数 vPCnum−2 が考えられる。これは 2PC を並列化可能ループ以外の MT の実行に割当てるということである。このように分割数を減らしていく上ですべての場合を考えるのは効率が悪いので、vPCnum−$Para^{k+1}_{i_1,...,i_k}$ を分割数の下限と定める。この値は、元からある並列実行可能な MT と分割されたループの MT が各 PC に 1 つずつ割当てられ Idle となる PC が存在しない状態を仮定した理想値である。

図 2 (a) に示す MTG(ここでは階層の表現は省略) の例を用いて説明する。例の MTG で、 (6, 1) のクラスタ構成すなわち 6PC,1PE の場合の分割数を計算すると、CP 上にある $MT_3$ は仮想 PC 数の 6 に、CP 上に無い $MT_5$,$MT_7$,$MT_8$ は、仮想 PC 数 −1 の 5 となる。その時の分割後の MTG を図 2 (b) に示し、実行時の各 PC への MT の割当て情報を図 3 に示す。図 2 (b) の形に分割された並列実行可能ループが図 3 に示されるように、シーケンシャルループ ($MT_9$) と並列に実行され、PC の Idle 時間が極めて小さくなってことが分かる。

#### 3.2.2 スケジュール長に基づく分割数の決定

本手法において $t_{diff} \gg 0$ すなわち vPCnum に比べて並列性が高い場合は、CP 上の MT を実行する PC には $t_{diff}$ に相当する PC の Idle 部分、つまりスケジューリングの空きが発生すると予想される。よって CP 上に無い並列化可能ループはその Idle 部分を埋める大きさに分割するのが望ましい。そこで本手法では $t_{diff}$ を出来るだけ効率よく埋めることの出来る大きさの MT に分割されるように分割数を各並列化可能ループ毎に求めて、さらにその値を vPCnum 数の倍数 に近似する手法等を用いて分割数を決定する。

例を挙げると、ある MTG を仮想 PC 数 4 でコストの算出を行う場合に $t_{diff}$ =100 であったとする。$MT_1$ のコストを仮に 1000 とすると $MT_1$ の分割数候補の一つは 10 となる。これを仮想 PC 数の 4 の倍数に近似すると 8,12 となる。この時 8,10,12 のうちどの分割数が有効であるか決定するために、それぞれの分割数候補で仮分割を行いコストの比較をする。そしてコストの最も小さくなる場合の分割数を仮想 PC 数 4 の場合の分割数として決定する。

### 3.3 クラスタリング決定における分割有効度

$MTG^{k+1}_{i_1,...,i_k}$ のクラスタリングの決定において、仮 PC 数 (vPCnum) と仮分割数が有効であるかを示す値として分割有効度 (Goodness) を以下のように定義する。

$$Goodness = t_{min\text{-}exe}/t_{schedule}$$

この値は平均プロセッサ占有率の予測値を示し、値が 1.0 の時 Idle となるプロセッサが無い状態を意味する。Goodness がある値 (ex. 0.8) 以上であればスケジューリング長の短縮において十分な効果が期待できるクラスタ構成及び分割数であると判断し、現在の仮想クラスタリング (vPCnum, vPEnum) と分割数を現階層の並列処理方式として決定する。また十分な効果が得られない場合は続けて他の分割数や他のクラスタリングを試行し比較を行う。

図 2 (a) に示す MTG において Para=2.05 である。ここで、この MTG に 6PE 割当てられた場合のクラスタリング毎の分割数候補をすべて試して Goodness を求めると表 1のようになる。クラスタリングは 3.1の手順 1で示したように (6,1) から試行し、分割数は 3.2.1 で決定したように (CP 上 6 分割、CP 外 5 分割) から試行するので、表 1の結果から一番始めに試行したクラスタリング及び分割数で最良の結果が得られ、提案するプロセッサクラスタリング決定手法が有効であることが分かる。

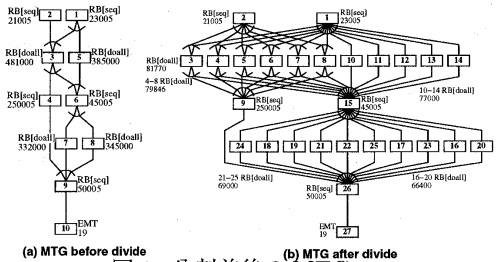


図 2: 分割前後の MTG

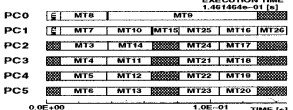


図 3: クラスタ (6,1) の場合の実行時の MT 割当て情報

表 1: 各クラスタリングにおける各分割数の分割有効度

| クラスタ (PC,PE) | 6,1 | 6,1 | 6,1 | 6,1 | 6,1 | 6,1 | 3,2 | 3,2 | 2,3 |
|---|---|---|---|---|---|---|---|---|---|
| CP 上の分割数 | 6 | 6 | 5 | 5 | 4 | 4 | 3 | 2 | 2 |
| CP 外の分割数 | 5 | 4 | 5 | 4 | 5 | 4 | 2 | 2 | 2 |
| 分割有効度 | .795 | .795 | .768 | .768 | .726 | .726 | .378 | .482 | .377 |
| 試行回数 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

## 4 まとめ

本稿ではマルチグレイン並列処理における階層的並列処理のためのプロセッサクラスタリング決定手法について述べた。

今後の課題は、本手法の SPEC、Perfect Club 等の実アプリケーションを用いた性能評価が挙げられる。

## 参考文献


[1] H.Kasahara, el al.: "A Multi-Grain Compilation Scheme for OSCAR", Proc. 4th Workshop on Languages and Compilers for Parallel Computing, Aug. 1991.

[2] Banerjee, U. : "Loop Transformations for Restructuring Compliers", Kluwer Academic Pub. 1993.

[3] Wolfe, M : "High Performance Compilers for Parallel Computing", Addison-Wesley Publishing Company. 1996.

[4] Carrie Brownhill, el al.: "Achieving Multi-level Parallelization", ISHPC. 1997.

[5] 本多, 他: "Fortran プログラム粗粒度タスク間の並列性の検出手法", 信学論, J73-D-I(12), Dec. 1991.

[6] H.Kasahara, el al.: "A Compilation Scheme for Macro-dataflow Computation on Hierarchical Multiprocessor Systems", Inter. Conf. on Parallel Processing, Aug. 1990.

[7] 笠原 博徳: "並列処理技術", コロナ社, Jun. 1991.

[8] M.W.Hall, el al.: "Maximizing Multiprocessor Performance with the SUIF Compiler", IEEE Computer, Dec. 1996.

[9] R.Eigenmann, el al.: "On the Automatic Parallelization of the Perfect Benchmarks", IEEE Transaction on parallel and distributed systems,Vol.9, Jan. 1998.

[10] 山本 晃正, 他: "OSCAR マルチグレイン並列化コンパイラにおける階層的並列処理手法", 情報処理学会第 58 回全国大会, 2D-04 Mar. 1999.