階層型粗粒度タスク並列処理のための タスク多重割当てを用いた実行方式

吉 田 明 正†

本論文では，マルチプロセッサシステム上での階層型マクロデータフロー処理において，粗粒度タスクの多重割当てを可能にする実行方式を提案する．階層型マクロデータフロー処理では，各階層の粗粒度タスク間並列性をコンパイラが自動抽出し，粗粒度タスクは複数プロセッサから構成されるプロセッサクラスタ（PC）に実行時に割り当てられて並列処理される．この際，従来手法では各粗粒度タスクは 1PC を専有して実行されており，対象プログラムの持つ各階層の並列性によっては PC 利用率が低くなることがある．そこで，本論文では，複数の粗粒度タスクを同一 PC に多重に割り当て，PC 利用率を向上させるタスク多重割当て実行方式，および，そのダイナミックスケジューリング手法を提案する．また，シミュレーションおよび SMP 上で行った性能評価の結果から，本手法により処理時間が顕著に短縮されることが確認された．

Execution Scheme Using Task Overlapping Assignment for Hierarchical Coarse Grain Task Parallel Processing

Akimasa Yoshida†

This paper proposes an execution scheme to enable overlapping assignment of coarse-grain tasks in hierarchical macro-dataflow processing on multiprocessor systems. In hierarchical macro-dataflow processing, a compiler extracts parallelism among coarse-grain tasks automatically and coarse-grain tasks are assigned to processor-clusters (PCs) at run-time. However, several programs may decrease PC utilization factor owing to lack of parallelism inside each coarse-grain task. Therefore, in order to improve PC utilization factor, this paper proposes an execution scheme with task overlapping assignment and its dynamic scheduling scheme to assign several coarse-grain tasks to a PC simultaneously. Also, the performance evaluations by simulations and executions on SMP showed the proposed scheme can reduce execution time remarkably.

1. はじめに

マルチプロセッサシステム上での自動並列化コンパイラを用いた並列処理では従来よりループ並列化手法[1)]が用いられており，たとえば，イリノイ大学の Polaris[2),3)]やスタンフォード大学の SUIF[4)]等のような先端の並列化コンパイラでは，強力なデータ依存解析手法とループリストラクチャリング手法を組み合わせることによりさまざまな形状のループが並列化可能になっている．しかしながら，これらのループ並列化手法では，単一ループのイタレーション間の並列性しか利用することができず，コストの大きいシーケンシャルループあるいは回転数の小さい並列ループを多く持つプログラムでは効果的な並列処理が行えないという問題があった．この問題を解決するために，ループやサブルーチン等の粗粒度タスクレベルの並列性を利用する粗粒度タスク並列処理[5),6)]，および階層的に粗粒度タスク並列処理を行う階層型マクロデータフロー処理[7),8)]が有効と考えられる．

階層型マクロデータフロー処理では，各階層の粗粒度タスク間並列性をコンパイラが自動抽出し，粗粒度タスクは複数プロセッサから構成されるプロセッサクラスタ（PC）に実行時に割り当てられて並列処理される．この際，従来手法[7),8)]では，粗粒度タスク内部のサブ粗粒度タスクのスケジューリングは，粗粒度タスクごとに独立に行われていたため，各粗粒度タスク（その内部にサブ粗粒度タスクを含む）は 1PC を専有して実行されていた．それゆえ，対象プログラムの持つ各階層の並列性によっては PC 利用率が低くなる可能性がある．

そこで，本論文では，複数の粗粒度タスクを同一 PC

† 東邦大学理学部情報科学科
Department of Information Science, Toho University

に多重に割り当て，それらの粗粒度タスク内部のサブ粗粒度タスク集合を並列実行することにより，PC 利用率を向上させ，対象プログラムの並列処理時間を短縮するタスク多重割当てを用いた実行方式を提案する．

本論文の構成は以下のとおりとする．2 章では階層型マクロデータフロー処理の概要を述べる．3 章では提案するタスク多重割当て実行の概念を述べ，4 章ではタスク多重割当てを用いたダイナミックスケジューリングの実現方式について述べる．5 章ではタスク多重割当て実行のシミュレーションによる性能評価，6 章では SMP 上でのマルチスレッドを用いた実装の性能評価，7 章では SPECfp95 ベンチマークプログラム turb3d を用いた性能評価について述べる．

2. 階層型マクロデータフロー処理

本章では，提案手法の対象となる階層型マクロデータフロー処理について述べる．

2.1 対象マルチプロセッサアーキテクチャ

階層型マクロデータフロー処理では，各プロセッサがインターコネクションネットワークを介して接続されている共有メモリ型マルチプロセッサシステムを対象とする．たとえば，商用 SMP（Symmetric Multiprocessor）上において実現する場合，階層型マクロデータフロー処理用の並列化コンパイラにより，OpenMP のようなスレッド命令を含む並列処理用プログラムを生成し，そのプログラムからマシンコードを商用コンパイラで生成することにより並列実行可能である[8]．

階層型マクロデータフロー処理を適用する場合，マルチプロセッサシステムを構成する全プロセッサ（PE）を，複数のグループに階層的に分割し，階層的にプロセッサクラスタを定義する．具体的には，マルチプロセッサシステム全体を第 0 階層プロセッサクラスタ（PC）と定義し，第 0 階層 PC 内の PE を複数のグループに分割し，それぞれを第 1 階層 PC と定義する．同様に，第 n 階層 PC 内の PE を複数のグループに分け第 $(n+1)$ 階層 PC を定義する．ただし，最下位の階層では PC が 1PE から構成される場合もある．

2.2 階層型マクロデータフロー処理のコンパイル手法

階層型マクロデータフロー処理[7]では，ループやサブルーチン等の粗粒度タスク（マクロタスク）が，プロセッサクラスタ（PC）に割り当てられて並列処理される．このとき，PC に割り当てられたマクロタスク内部では，階層的に粗粒度タスクレベルの並列処理が適用される．以下では，この階層型マクロデータフロー処理のコンパイル手法について述べる．

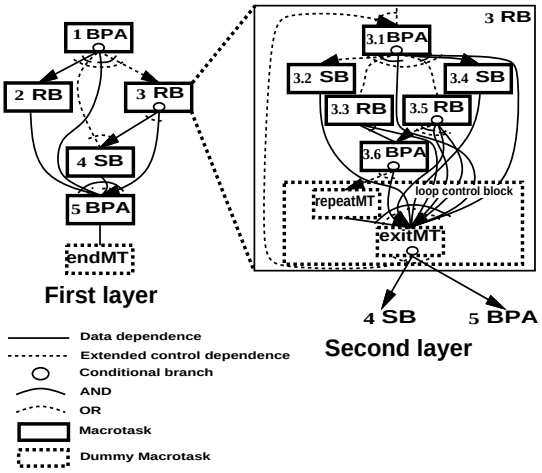

図 1 階層型マクロタスクグラフ（MTG）
Fig. 1 Hierarchical macrotask graph (MTG).

2.2.1 階層型マクロタスク（MT）生成

階層型マクロデータフロー処理手法では，まず，プログラム（全体を第 0 階層マクロタスクとする）を第 1 階層マクロタスクに分割する．マクロタスク（MT）は，擬似代入文ブロック（BPA），繰返しブロック（RB），あるいは，サブルーチンブロック（SB）の 3 種類から構成される[7]．

次に，第 1 階層マクロタスク（RB または SB）の内部に複数のサブマクロタスク（サブ BPA，サブ RB，サブ SB）を含んでいる場合には，それらのサブマクロタスクを第 2 階層マクロタスクとして定義する．

2.2.2 階層型マクロタスクグラフ（MTG）生成

マクロタスク生成後，マクロタスク間あるいはサブマクロタスク間のコントロールフローとデータフローを解析し，階層型マクロフローグラフ[7]を生成する．

次に，コントロール依存とデータ依存を考慮したマクロタスク間並列性を最大限に引き出すために，各マクロタスクの最早実行可能条件[7]を解析する．最早実行可能条件は，コントロール依存とデータ依存を考慮したマクロタスク間の並列性を最大限に表している．この各マクロタスクの最早実行可能条件は，**図 1** のような階層型マクロタスクグラフ（MTG）[7]で表すことができる．

2.2.3 階層型ダイナミックスケジューリングルーチン生成

階層型マクロデータフロー処理では，マクロタスク間の条件分岐等の実行時不確定性に対処するために，マクロタスクを実行時にプロセッサクラスタ（PC）に割り当てるダイナミックスケジューリング方式を採用している．また，コンパイラが生成するダイナミッ

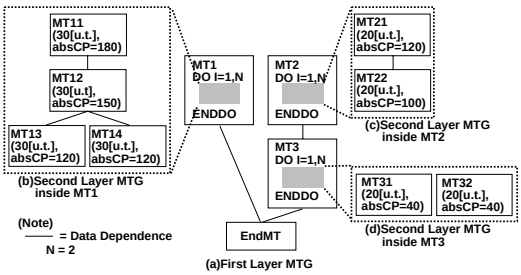

図 2 サンプルプログラムの MTG
Fig. 2 MTG of a sample program.

クスケジューリングルーチンを用いることにより，低オーバヘッドでマクロタスクの割当てを行うことが可能である．ダイナミックスケジューリングの基本的なアルゴリズムとしては，Dynamic-CP 法を使用している[7),9)]．なお，本論文で提案するタスク多重割当てをともなうダイナミックスケジューリングを用いる場合には，4 章で述べる方式を採用する．

3. タスク多重割当て実行の概念

本章では，提案するタスク多重割当てをともなう階層型マクロデータフロー処理の概念を述べる．従来の階層型マクロデータフロー処理（タスク多重割当てなし）では，1 つのプロセッサクラスタ（PC）には 1 つのマクロタスク（MT）が割り当てられるが，提案手法では 1PC に同時に 2 つ以上マクロタスクを割り当てることが可能である．

タスク多重割当てを用いない従来の階層型マクロデータフロー処理では，**図 2** のプログラムを，第 1 階層が 1PC で第 2 階層が 2PE のマルチプロセッサシステム上で並列処理すると，**図 3** (a) のような実行イメージになる．この場合，MT1 内の MT13 と MT14 間，および MT3 内の MT31 と MT32 間の並列性は引き出されているが，PE1 の黒い部分はアイドル状態となっており，PC 内 PE が有効に利用されていないことが分かる．各 MT における第 1 階層 PC（2PE から構成される）の利用率は，MT1 は 67%であり，MT2 は 50%，MT3 は 100%である．なお，第 1 階層 MT（たとえば MT1）は 2 回転のループであるため，その内側のサブ MT（たとえば MT11〜MT14）はそれぞれ 2 回ずつ実行されている．

次に，本論文で提案するタスク多重割当てを用いた階層型マクロデータフロー処理の場合には，同じく第 1 階層が 1PC で第 2 階層が 2PE のマルチプロセッサシステム上で並列処理すると，図 3 (b) のように，MT2 あるいは MT3 が MT1 と同時に同一の第 1 階層 PC（2PE 構成）に割り当てられる．すなわち，MT1,

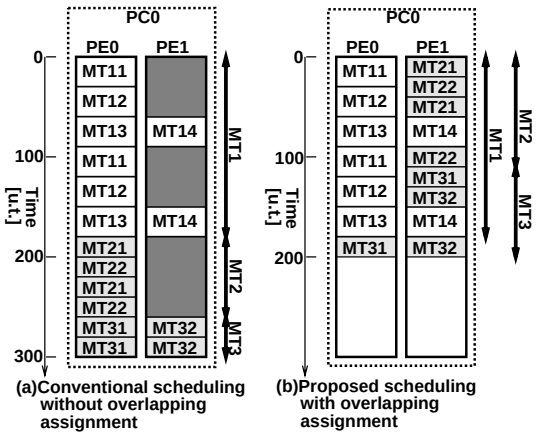

図 3 サンプルプログラムの実行イメージ
Fig. 3 Execution image of a sample program.

MT2，MT3 のサブ MT 間の並列性を効果的に利用することにより，PC 利用率は約 100%になり実行時間が大幅に短縮される．

上記の例からも分かるように，従来の実行方式では，MT 内のサブ MT 間並列性が少ない場合には，サブ MT は PC 内の一部の PE で実行されることになり，最悪の場合の PC 利用率は1/(PC 内 PE 数) になる．それに対して，提案手法では，ある MT の PC 利用率が低い（すなわちその MT 内部のサブ MT 間並列性が少ない）場合には，その MT を実行している PC に他 MT を多重に割り当てることにより，PC 利用率を 100%に近づけることが可能となる．

4. タスク多重割当てを用いたダイナミックスケジューリング実現方式

本章では，3 章で述べたタスク多重割当てをともなうダイナミックスケジューリングの実現方式について述べる．タスク多重割当てをともなうダイナミックスケジューリング手法は，多重割当てをともなう上位階層ダイナミックスケジューリングと，多重割当てされた MT 内のサブ MT のスケジューリングを行う下位階層ダイナミックスケジューリングにより実現される．

4.1 MT 内サブ MT の絶対 CP 長

タスク多重割当てをともなわない階層型マクロデータフロー処理では，MT 内のサブ MT のスケジューリングは各 MT ごとに独立に行われていたため，スケジューリングのプライオリティとして用いられるサブ MT の CP 長（Ciritical Path）[9)]の値は，MT 内のサブ MTG ごとに計算された相対的な CP 長が用いられていた．しかしながら，相対的な CP 長は，本論文で提案するタスク多重割当てのように，異なる MT 内

のサブ MT を統一的に管理する場合には必ずしも有効ではない．

そこで，本手法では，各 MT_i 内の サブ MT_j の CP 長は，その MT_i 内の サブ MT_j が最初に実行されるときの絶対的な CP 長（本論文では絶対 CP 長と呼ぶ）とする．本手法で定義する絶対 CP 長（CP^{abs}_{subMTj}）は，

$$CP_{MTi} - CP_{subMTGi} + CP^{rel}_{subMTj}$$

により求められる．ここで，CP_{MTi} は上位階層 MTG における出口ノードから MT_i までの CP 長とし，$CP_{subMTGi}$ は MT_i 内のサブ MTG 単体の CP 長，CP^{rel}_{subMTj} は MT_i 内のサブ MTG における サブ MT_j の相対的な CP 長とする．たとえば，図 2 の MT1 内の MT11 の絶対 CP 長は，$CP_{MT1} = 180[u.t.]$，$CP_{subMTG1} = 90[u.t.]$，$CP^{rel}_{subMT11} = 90[u.t.]$ であり，$CP^{abs}_{subMT11} = 180 - 90 + 90 = 180[u.t.]$ となる．同様に各サブ MT の絶対 CP 長は，図 2 の absCP に書かれた値となる．

4.2 MT の PC 利用率

本ダイナミックスケジューリング手法では，MT を PC（プロセッサクラスタ）に割り当てる際に，各 MT の PC 利用率を用いる．各 MT_i の PC 利用率は，

$$\frac{T^{seq}_{MTi}}{T^{para}_{MTi} * N_{PEinPC}}$$

により求められる．ここで，T^{seq}_{MTi} は MT_i のシーケンシャル処理時間とし，T^{para}_{MTi} は MT_i の PC 内 PE 上での並列処理時間，N_{PEinPC} は PC 内 PE 数とする．本手法では，各 MT の PC 利用率をスタティックに求めるため，PC 利用率計算の際には，T^{seq}_{MTi} は MT_i 内サブ MTG を構成するサブ MT の処理時間の合計とし，T^{para}_{MTi} は MT_i 内のサブ MTG 単体の CP 長としている．ただし，この方法で求めた PC 利用率が 1.0 を超える場合には，1.0 として再定義する．

4.3 PC 単位レディサブ MT キュー

階層型マクロデータフロー処理においてダイナミックスケジューリングを行う場合，各 MT はその最早実行可能条件[7]が満たされた後，レディMT キューに投入され，プライオリティの高い（CP 長の大きい）MT から順に PC に割り当てられる．ここで，従来のようにタスク多重割当てを用いない場合，MT 内のサブ MT のダイナミックスケジューリングは，その MT 単位で独立に行えばよいため，MT ごとに用意したレディサブ MT キューを使用していた．

それに対して，本論文で提案するタスク多重割当てをともなうダイナミックスケジューリングを実現する場合，同一 PC に多重割当てされた異なる MT のサブ MT を統一的に扱うため，従来のような MT ごとのレディサブ MT キューでは実現できない．そこで，本手法では，各 PC ごとに用意したレディサブ MT キューを使用する．したがって，ある PC に多重割当てされた MT のサブ MT は，その PC に対して用意されたレディサブ MT キューに，最早実行可能条件が満たされた後，投入される．

4.4 上位階層ダイナミックスケジューリング

上位階層ダイナミックスケジューリング（MT 用）では，最早実行可能条件を満たす MT の中で CP 長の大きい MT から順に PC に割当てを行う．この際，本スケジューリング処理部では，PC 利用率が最小の PC（PC 利用率は 100%未満とする）を選び，MT をその PC に割り当てる．なお，各 PC の PC 利用率は，当該 PC に現在割り当てられている全 MT の PC 利用率の合計であり，この値の更新は，PC に MT を割り当てる際にその MT の PC 利用率を加算し，MT の実行終了時にその MT の PC 利用率を減算することにより行われている．なお，アイドル PC の PC 利用率は 0%として扱う．

以下に上位階層ダイナミックスケジューリングの具体的な手順を示す．

(i) 上位階層 MTG 上において最早実行可能条件を満たす MT を，上位階層用レディMT キューに投入する．

(ii) 上位階層用レディMT キューを，MT の CP 長順にソートする．

(iii) 上位階層用レディMT キューから先頭（CP 長が最大）の MT を取り出し，最小の PC 利用率の PC（利用率は 100%未満とする）に割り当てる．

4.5 下位階層ダイナミックスケジューリング

下位階層ダイナミックスケジューリング（サブ MT 用）では，上位階層ダイナミックスケジューリングによって MT の割り当てられた PC_p 内の PE に，その MT 内のサブ MT を割り当てる．この際，PC_p のために用意されたレディサブ MT キューを用いる．

具体的な手順は，各 PC_p に対して以下のとおりである．

(i) PC_p に割り当てられている各 MT_i 内部において，最早実行可能条件を満たすサブ MT を，PC_p 用レディサブ MT キューに投入する．

(ii) PC_p 用レディサブ MT キューを，サブ MT の絶対 CP 長順にソートする．

(iii) PC_p 用レディサブ MT キューから先頭（絶対

CP 長が最大）のサブ MT を取り出し，PC_p 内のアイドル PE に割り当てる．

4.6 マルチスレッドによる実装方法

階層型マクロデータフロー処理の並列処理用プログラムを，スレッド命令を用いて実現する場合，分散型ダイナミックスケジューリング方式と集中型ダイナミックスケジューリング方式の実装が可能である[8),10)]．

タスク多重割当てをともなう分散型ダイナミックスケジューリング方式では，PE 数分のスレッドコードを用意し，各スレッドコードは，上位階層ダイナミックスケジューリング処理部，下位階層ダイナミックスケジューリング処理部，マクロタスク処理部から構成される．これらのスレッドは実行開始時に 1 回のみ生成される．本スレッドコードにより，各 PE 上でマクロタスクの実行を終えるたびに，ダイナミックスケジューリング処理部を呼び出してスケジューリングを行い，新たに割り当てられたマクロタスクを実行する．なお，レディマクロタスクキューのアクセスに対しては排他制御を行う．

一方，タスク多重割当てをともなう集中型ダイナミックスケジューリング方式では，1PE がスケジューリング処理専用となるため，スケジューリング処理用 PE のスレッドコードは上位階層ダイナミックスケジューリング処理部と下位階層ダイナミックスケジューリング処理部から構成され，マクロタスク処理用 PE のスレッドコードは，マクロタスク処理部から構成される．

5. タスク多重割当て実行のシミュレーションによる性能評価

本章では，提案するタスク多重割当てをともなう階層型マクロデータフロー処理を，シミュレーションにより性能評価する．

5.1 評価用マクロタスクグラフ

本シュミレーションでは，**表 1** に示す各カテゴリ（I-i から III-iii）ごとに 10 個のマクロタスクグラフ（MTG）を用意し，計 90 個の MTG を用いて性能評価を行った．各カテゴリごとに上位階層の MT 間並列性と下位階層（MT 内）のサブ MT 間並列性は異なっており，カテゴリを示す記号 I，II，III は，上位階層の並列性を小さい順に表しており，一方，記号 i，ii，iii は，下位階層の並列性を小さい順に表している．たとえば，カテゴリ III-i では，上位階層の MT 間並列性は大きく，下位階層のサブ MT 間並列性は小さい分類である．

このランダム MTG の生成は，文献 11) で行われているように，タスクグラフの *Layer*数 を決定し，各 Layer において *Width* 数 の MT を配置し，異なる Layer 上の MT 間においてデータ依存エッジを張ることにより行っている．ここで使用したパラメータは，表 1 の上位階層のパラメータに示すとおりである．なお，カテゴリ I，II，III における上位階層の並列性の違いは，*Width* 数 の範囲を調節することにより行われている．

次に，下位階層（MT 内）のサブ MTG の並列性に関しては，表 1 の下位階層のパラメータのとおり，サブ MTG を構成するサブ MT の処理時間の合計に対するサブ MTG 上の CP 長（Critical Path）の割合（$CPinSubMTG$）と，サブ MTG 上の CP におけるループ並列処理不可能部分の割合（$SeqinCP$）により決定している．

ここで，$CPinSubMTG$ の値が 1.0 に近いということは，サブ MTG を構成するサブ MT のほとんどが CP 上に存在し，サブ MT 間の粗粒度並列性が非常に小さいことを意味している．また，$SeqinCP$ の値は，サブ MTG の CP 上のサブ MT においてループ並列処理のできない（シーケンシャル処理される）割合を表している．すなわち，ループ並列処理可能部分の処理時間は1/(PC 内 PE 数) に短縮されるため，サブ MTG

表 1 評価用マクロタスクグラフのカテゴリ分類
Table 1 Classification of macrotask graphs for evaluations.

MTG のカテゴリ	上位階層のパラメータ				下位階層のパラメータ		並列性	
	Width 数	Layer 数	MT の先行エッジ数	MT の処理時間	CPinSubMTG	SeqinCP	上位階層	下位階層
I-i	1–2	20	4–6	100–1000 [u.t.]	0.8–0.9	0.8–0.9	小	小
I-ii					0.4–0.5	0.8–0.9		中
I-iii					0.4–0.5	0.4–0.5		大
II-i	1–4	20	4–6	100–1000 [u.t.]	0.8–0.9	0.8–0.9	中	小
II-ii					0.4–0.5	0.8–0.9		中
II-iii					0.4–0.5	0.4–0.5		大
III-i	1–6	20	4–6	100–1000 [u.t.]	0.8–0.9	0.8–0.9	大	小
III-ii					0.4–0.5	0.8–0.9		中
III-iii					0.4–0.5	0.4–0.5		大

の CP 長は，PC 内 PE 数に依存して，($SeqinCP$+(1-$SeqinCP$)/(PC 内 PE 数)) 倍に短縮される．たとえば，$SeqinCP = 0.4$ で PC 内 PE 数=16 の場合は，0.4+(1 − 0.4)/16=0.44 倍にサブ MTG の CP 長が短縮される．

5.2 シミュレーションの方法

本シミュレーションでは，前節で生成された MTG 情報を入力とし，その MTG 上の各 MT を，シミュレータ上のプロセッサクラスタ（PC）に多重割当てを考慮してスケジューリングし，階層型マクロデータフロー処理による実行時間を求めている．ここで，PC 上での各 MT の並列処理時間は，PC を構成する PE 数に応じて変化するため，以下の手順で求める．PC 内 PE 数が N_{PEinPC} のときの MT_i の並列処理時間 T^{para}_{MTi} は，MT_i のシーケンシャル処理時間 T^{seq}_{MTi} と表 1 の下位階層パラメータを用いて，$T^{seq}_{MTi} * CPinSubMTG * (SeqinCP + (1 - SeqinCP)/N_{PEinPC})$ として表される．言い換えると，この値は MT_i 内のサブ MTG の CP 上を実行したときの並列処理時間となる．ただし，この値が，T^{seq}_{MTi}/N_{PEinPC} より小さい場合には，MT の並列処理時間 T^{para}_{MTi} として，代わりに T^{seq}_{MTi}/N_{PEinPC} を用いる．

また，シミュレーションにおいて，PC に複数の MT が多重割当てされた場合には，CP 長の大きい MT から順に，その MT が要求した PC 利用率を確保できるようにしている．これは，CP 長の大きい MT 内のサブ MT の最大絶対 CP 長は，CP 長の小さい MT 内のサブ MT の最大絶対 CP 長より大きいからである．このとき，要求した PC 利用率を確保できなかった MT の当該 PC における並列処理時間は，要求した PC 利用率を確保できた場合の並列処理時間に，(要求した PC 利用率)/(確保できた PC 利用率) を乗じたものとしている．

5.3 各クラスタ構成における実行結果

本節では，5.1 節で生成された MTG に対して，さまざまなプロセッサのクラスタ構成上でシミュレーション実行した結果を**表 2** に示す．ここで，クラスタ構成は，16PE で実現可能な 1PC*16PE（PC は 16PE），2PC*8PE（各 PC は 8PE），4PC*4PE（各 PC は 4PE），8PC*2PE（各 PC は 4PE），16PC*1PE（各 PC は 1PE）の 5 通りである．この表では，各カテゴリごとに，各クラスタ構成上での 1PE に対する速度向上率（各カテゴリでは 10 個の MTG の平均）を示している．各行の上段はタスク多重割当てなしの実行の場合であり，下段はタスク多重割当てありの実行の場合である．また，表 2 の最下行は，各クラスタ構成ごとに，タスク多重割当て実行による速度向上率（全カテゴリの 90 個の MTG の平均）を示している．

表 2 各クラスタ構成での速度向上率（16PE 上）
Table 2 Speedup ratio in each processor clustering on 16PE.

カテゴリ	多重	1PC*16PE	2PC*8PE	4PC*4PE	8PC*2PE	16PC*1PE
I-i	無	1.37	2.46	2.90	2.80	2.20
	有	3.01	2.97	2.91	2.80	
I-ii	無	2.58	4.63	5.49	4.40	2.20
	有	5.68	5.62	5.51	4.40	
I-iii	無	4.56	7.75	8.20	4.40	2.20
	有	9.77	9.20	8.07	4.40	
II-i	無	1.37	2.66	4.39	4.50	3.55
	有	4.86	4.80	4.70	4.50	
II-ii	無	2.60	5.03	8.27	7.10	3.55
	有	9.11	9.01	8.56	7.10	
II-iii	無	4.62	8.44	12.40	7.10	3.55
	有	13.61	13.36	12.08	7.10	
III-i	無	1.37	2.68	4.93	5.84	4.59
	有	6.29	6.22	6.09	5.80	
III-ii	無	2.58	5.07	9.31	9.18	4.59
	有	11.54	11.43	10.59	9.18	
III-iii	無	4.56	8.47	13.88	9.18	4.59
	有	14.84	14.91	13.80	9.18	
多重向上率		220%	67%	4%	0%	

全体的に PC を構成する PE 数が増えると，PC の利用率が低くなり，かつ，多重割当てされる MT 数が増えるため，タスク多重割当ての効果が最大限に発揮され良い性能が得られている，たとえば，1PC*16PE 構成の場合には，タスク多重割当てなしの場合と比べて 220%の速度向上が得られることが分かる．一方，8PC*2PE 構成では，対象 MTG の MT 間並列性が十分にないため，多重割当てによる速度向上はなかった．

また，1PC*16PE のクラスタ構成においてカテゴリ単位でみると，III-i のように上位階層の並列性が大きく，下位階層の並列性が小さい場合には，多重割当てによって従来の 1.37 倍から 6.29 倍へと顕著な速度向上がみられる．

以上の結果，従来のタスク多重割当てなしの場合には，対象プログラム（MTG）が持つ上位・下位階層の並列性を，クラスタ構成によっては十分に利用できないが，タスク多重割当てを適用した場合にはどのクラスタ構成においても，安定して対象プログラム（MTG）が持つ上位・下位階層の並列性を十分に利用できることが分かる．特に，PC を構成する PE が多い場合には良い結果を示すことが分かり，大規模なシステム上において提案手法は有効であると考えられる．

5.4 最善クラスタ構成における実行結果

前節では，各クラスタ構成ごとにタスク多重割当ての有無の性能を比較していたのに対して，本節では，

表 3 最善クラスタ構成での速度向上率（16PE 上）
Table 3 Speedup ratio in best processor clustering on 16PE.

カテゴリ	多重なし	多重あり	多重による向上率		
			平均	最小	最大
I-i	2.91	3.01	3.5%	1.2%	7.7%
I-ii	5.50	5.68	3.4%	1.1%	6.8%
I-iii	8.44	9.77	15.6%	6.9%	21.3%
II-i	4.56	4.86	6.4%	3.4%	9.0%
II-ii	8.27	9.11	10.1%	3.3%	18.2%
II-iii	12.40	13.61	9.9%	7.1%	19.7%
III-i	5.84	6.29	7.6%	6.3%	8.3%
III-ii	9.77	11.54	17.8%	9.9%	30.1%
III-iii	13.88	14.96	7.9%	5.2%	12.2%

各プログラム（MTG）ごとに最善のクラスタ構成（タスク多重割当てなしとありの場合で最善クラスタ構成が異なることがある）を利用した場合のタスク多重割当ての有無による性能比較を行う．

結果は**表 3** に示すとおりであり，この表では各プログラム（MTG）ごとに，タスク多重割当てなしとありのそれぞれにおいて，最善のクラスタ構成（16PE 実行の中で最小の実行時間を与えるクラスタ構成）を選択した場合の 1PE に対する速度向上率を，カテゴリごとに平均化したものである．また，多重による向上率では，各カテゴリの 10 個の MTG における平均，最小，最大を示している．

たとえば，従来のタスク多重割当てなしでは，表 3 のカテゴリ III-ii の場合，最善のクラスタ構成を採用しても 9.77 倍の速度向上しか得られないのに対して，タスク多重割当てありでは 11.54 倍の速度向上が得られており，タスク多重割当てにより平均 17.8%（最大 30.1%）の速度向上が得られることが分かる．

また，提案するタスク多重割当て実行は，対象プログラム（MTG）の上位・下位階層の並列性を最大限に利用することにより，全カテゴリにおいて速度向上が得られており，タスク多重割当ての有効性が確認できた．

6. SMP 上でのマルチスレッドを用いた実装の性能評価

本章では，タスク多重割当てを用いたダイナミックスケジューリングをともなう階層型マクロデータフロー処理を，SMP（Sun Enterprise 3000）上で性能評価した結果について述べる．

6.1 SMP 上での性能評価環境

性能評価に用いる Sun Enterprise 3000 は，Ultra SPARC（250 MHz）プロセッサ 4 台を，Gigaplane インターコネクトにより，256 MB の集中共有メモリに接続した SMP マシンである．OS は Solaris2.6 であり，コンパイラは Sun Workshop Compilers4.2 の C コンパイラを使用している．

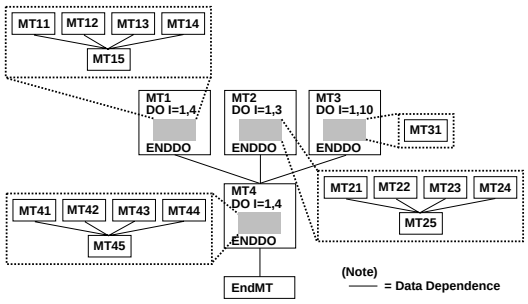

図 4 実装評価用プログラムの MTG
Fig. 4 MTG of a program for implementation evaluation.

表 4 実装評価用プログラムの SMP 上での実行結果
Table 4 Execution result of a program for implementation evaluation on SMP.

全 PE 数	クラスタ構成	多重割当て	実行時間 [ms]	速度向上率 (1PE 比)
1	1PC*1PE	無	139.571	1.00
4	1PC*4PE	無	72.011	1.94
4	2PC*2PE	無	68.434	2.04
4	4PC*1PE	無	84.673	1.65
4	1PC*4PE	有	45.829 (48.496†)	3.05 (2.88†)
4	2PC*2PE	有	53.979 (59.274†)	2.59 (2.35†)

†：多重割当て時に絶対 CP 長を適用しなかった場合．

本評価では，**図 4** のプログラムに対して，提案手法をともなう階層型マクロデータフロー処理の並列処理用プログラムを，Solaris スレッドを用いて C 言語により実装した．この際，4.6 節で述べた分散型ダイナミックスケジューリング方式と集中型ダイナミックスケジューリング方式の実装が考えられるが，本評価で用いる SMP はプロセッサ数が 4 と少ないため，スケジューリング処理用専用プロセッサの不要な分散型ダイナミックスケジューリング方式を採用した．

6.2 マルチスレッドによる実装の性能評価

図 4 の実装評価用プログラムは，第 1 階層が 4 つの MT（マクロタスク）から構成されており，各 MT の内部（第 2 階層）にサブ MT が存在する．MTG 情報として，各サブ MT の処理時間は 100 [u.t.] としており，第 1 階層の各 MT の PC 利用率は PC を 4PE 構成とする場合，MT1 は 63%，MT2 は 63%，MT3 は 25%，MT4 は 63%となる．

このプログラムを Enterprise 3000 上で実行した結果を**表 4** に示す．まず，従来のタスク多重割当てなしの階層型マクロデータフロー処理による 4PE 上での実

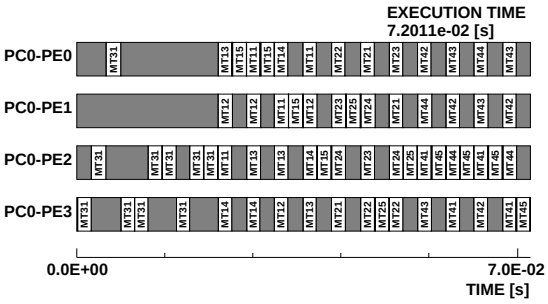

図 5 多重割当てなしの実行トレース（1PC*4PE）
Fig. 5 Execution trace without overlapping assignment (1PC*4PE).

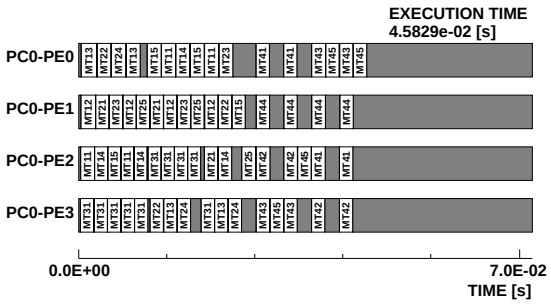

図 6 多重割当てありの実行トレース（1PC*4PE）
Fig. 6 Execution trace with overlapping assignment (1PC*4PE).

行時間は，1PC*4PE クラスタ構成（PC は 4PE）の場合，1PE の処理時間の 1/1.94 に短縮され，2PC*2PE（各 PC は 2PE）の場合は 1PE の 1/2.04，4PC*1PE（各 PC は 1PE）の場合は 1PE の 1/1.65 に短縮される．すなわち，このプログラムでは，2PC*2PE 実行が多重割当てなしの場合に最小の実行時間を与える．ここで，1PC*4PE クラスタ構成の実行トレースを**図 5** に示す．本トレースでは，CP 長の高い MT3，MT1，MT2，MT4 の順に PC0（PC0 の PE0〜PE3）に割り当てられており，その各 MT 内のサブマクロタスクが，PC 内のどの PE 上で実行されるかを表している．なお，MT1 内の MT11〜MT15，MT2 内の MT21〜MT25，MT3 内の MT31，MT4 内の MT41〜MT45 は，それぞれ MT1，MT2，MT3，MT4 のループの回転数分実行されている．この実行トレースから，多重割当てを適用しない場合には PC 利用率が低いことが分かる．

次に，提案するタスク多重割当てをともなうダイナミックスケジューリングを用いた階層型マクロデータフロー処理による 4PE 上の実行時間は，1PC*4PE クラスタ構成（PC は 4PE）の場合，1PE の処理時間の 1/3.05 に短縮され，2PC*2PE（各 PC は 2PE）の場合は 1PE の 1/2.59 に短縮される．すなわち，このプログラムでは，1PC*4PE 実行が多重割当てありの場合に最小の実行時間を与える．

1PC*4PE クラスタ構成の実行トレースは**図 6** のとおりであり，サブマクロタスクが PC 内の各 PE 上で効率良く実行されていることが分かる．たとえば，実行開始時には，MT1，MT2，MT3 が PC0（PC0 の PE0〜PE3）に多重に割り当てられており，この段階で PC0 用のレディサブ MT キューには，MT11〜MT14，MT21〜MT24，MT31 が投入されており，絶対 CP 長の高い MT31 が優先的に PE に割り当てられる．この際，絶対 CP 長を適用しなかった場合には，表 4 の†印で示したように性能が低下することが分かる．また，本評価において各サブマクロタスクの処理内容は，フィボナッチ数列を 20 項まで求める計算であり，その処理時間との比較からも，タスク多重割当てをともなうダイナミックスケジューリングのオーバヘッドはきわめて小さいことが分かる．

以上の点から，タスク多重割当てありのダイナミックスケジューリングによる 1PC*4PE 実行は，タスク多重割当てなしの 2PC*2PE 実行と比べて，実行時間が 33.0%短縮されており，タスク多重割当てをともなうダイナミックスケジューリングの有効性が確認され，また，低オーバヘッドで Solaris スレッドにより実装可能であることが確認された．

7. SPECfp95 ベンチマークプログラム turb3d を用いた性能評価

本章では，提案するタスク多重割当てをともなう階層型マクロデータフロー処理の性能評価を，SPECfp95 ベンチマークプログラム turb3d を用いて行う．turb3d プログラムは，2101 行の Fortran ソースコードから構成されており，ナビエストークス方程式を解くことにより乱気流のシミュレーションを行うことが可能である．このプログラムの実行時間の 99.4%を費やすメイン収束ループの MTG は，ループディストリビューション適用後，**図 7** のような形状をしており，62 個のマクロタスク（内部にはサブマクロタスクが含まれる）から構成される．図 7 の MTG のノード内番号は MT 番号を表している．

本性能評価では，まず，turb3d プログラムを 6 章で用いた Sun Enterprise 3000 上で F77 コンパイラでコンパイルしてシーケンシャル実行することにより，各マクロタスクの実行時間と条件分岐のプロファイルを取得する．なお，図 7 から構成されるメイン収束ループは 12 回実行されるため，それぞれに対

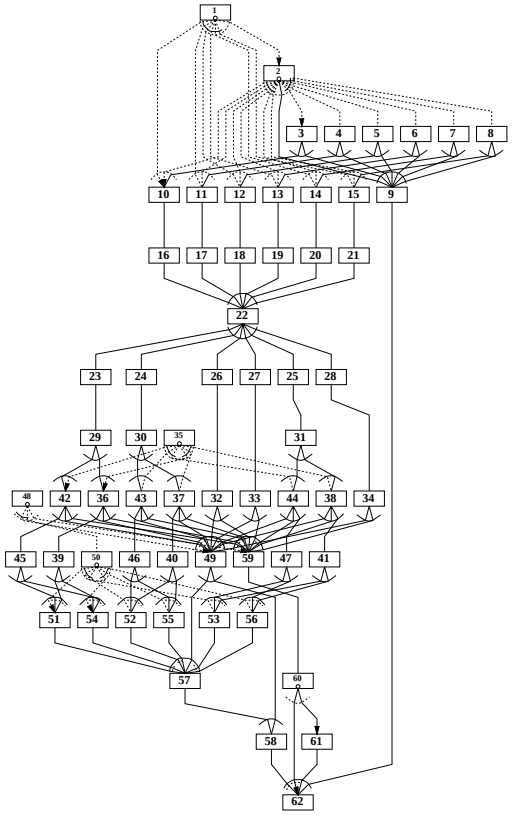

図 7 turb3d プログラムの MTG
Fig. 7 MTG of turb3d program.

してプロファイル情報を取得しておく．また，5.1 節で定義された各マクロタスク内部の並列性のパラメータ（$CPinSubMTG$ と $SeqinCP$）を解析すると，図 7 の MT3〜MT8，MT36〜MT47，MT49，MT51〜MT57，MT59 は，その内部においてサブマクロタスク間並列性はなく，ループ並列処理が可能であるため，$CPinSubMTG = 1.0$，$SeqinCP = 0.0$ となる．他の MT はその内部のサブマクロタスク間並列性はなく，ループ並列処理も不可能であるため，$CPinSubMTG = 1.0$，$SeqinCP = 1.0$ となる．

次に，上述の実行プロファイルと MTG 情報を用いて，5 章のシミュレーションにより各実行方式における並列処理時間を測定すると，**表 5** の結果が得られる．表 5 によると，4PE 上で最小の実行時間を与える実行方式は，多重割当てありの 1PC*4PE 実行であり，1PE の実行時間の 1/3.75 となっている．これは，多重割当てありの 4PC*1PE 実行の 1/3.03 と比べて 19.3%実行時間が短縮されている．また，8PE 上で最小の実行時間を与える実行方式は多重割当てありの 1PC*8PE 実行（1PE の 1/6.06）であり，多重割当てなしの 8PC*1PE 実行の 1/4.11 と比べて 32.2%実行時間が短縮されている．

表 5 turb3d プログラムの実行結果
Table 5 Execution result of turb3d program.

全 PE 数	クラスタ構成	多重割当ての有無	実行時間 [s]	速度向上率 (1PE 比)
1	1PC*1PE	無	61.313	1.00
4	1PC*4PE	無	52.971	1.16
4	2PC*2PE	無	29.306	2.09
4	4PC*1PE	無	20.249	3.03
4	1PC*4PE	有	16.342	3.75
4	2PC*2PE	有	17.487	3.51
8	1PC*8PE	無	51.581	1.19
8	2PC*4PE	無	27.327	2.24
8	4PC*2PE	無	17.002	3.61
8	8PC*1PE	無	14.907	4.11
8	1PC*8PE	有	10.120	6.06
8	2PC*4PE	有	10.707	5.73
8	4PC*2PE	有	12.042	5.09

以上まとめると，提案するタスク多重割当てをともなう実行方式は，turb3d ベンチマークプログラムにおいて，タスク多重割当てなしの場合と比べて 19.3%〜32.2%の実行時間短縮が可能であり，提案手法の有効性が確かめられた．

8. おわりに

本論文では，階層型マクロデータフロー処理において，粗粒度タスクの多重割当てを可能にするタスク多重割当てを用いた実行方式を提案した．本方式では，複数の粗粒度タスクを同一 PC（プロセッサクラスタ）に多重に割り当て，それらの粗粒度タスク内部のサブ粗粒度タスク集合を並列実行することにより，PC 利用率を向上させ，実行時間の短縮を可能にしている．

本手法をシミュレーションにより上位・下位階層の並列性の異なる 90 個のマクロタスクグラフに対して評価したところ，最大で 30%（特定のクラスタ構成では平均 220%）の性能向上が得られることが確認され，SMP 上で行った性能評価から，提案手法は低オーバヘッドでマルチスレッドにより実装可能であることが確認された．また，SPECfp95 ベンチマークプログラム turb3d による性能評価においても，8PE 実行の場合にタスク多重割当てにより 32.2%の実行時間短縮が確認されており，提案手法の有効性が確かめれた．

今後の課題としては，提案手法を自動並列化コンパイラに組み込み，さまざまなベンチマークプログラムで性能評価を行うことがあげられる．また，コンパイル時に同一 PC に多重割当てすべき粗粒度タスク集合を決定しておき，パーシャルスタティックタスク割当て[12]をともなうダイナミックスケジューリングと併

用することにより，それらの粗粒度タスク集合を同一PCに多重割当てすることも考えられる．

謝辞 本研究の一部は，科学研究費補助金（奨励研究（A）12780243），および，NEDOアドバンスト並列化コンパイラプロジェクトの一環として行われた．

参 考 文 献

1) Wolfe, M.: *High Performance Compilers for Parallel Computing*, Addison-Wesley Publishing Company (1996).
2) Blume, W., Doallo, R., Eigenmann, R., Grout, J., Hoeflinger, J., Lawrence, T., Lee, J., Padua, D., Paek, Y., Pottenger, B., Rauchwerger, L. and Tu, P.: Advanced Program Restructuring for High Performance Computers with Polaris, Technical Report 1473, CSRD, University of Illinois, Urbana-Champaign (1996).
3) Eigenmann, R., Hoeflinger, J. and Padua, D.: On the automatic parallelization of the Perfect benchmarks, *IEEE Trans. Parallel and Distributed System*, Vol.9, No.1 (1998).
4) Amarasinghe, S., Anderson, J., Lam, M. and Tseng, C.-W.: The SUIF Compiler for Scalable Parallel Machines, *Proc. 7th SIAM Conference on Parallel Processing for Scientific Computing* (1995).
5) Kasahara, H., Honda, H., Mogi, A., Ogura, A., Fujiwara, K. and Narita, S.: Multi-Grain Parallelizing Compilation Scheme for OSCAR, *4th Workshop on Language and Compilers for Parallel Computing* (1991).
6) Martorell, X., Ayguade, E., Navarro, N., Corbalan, J., Gonzalez, M. and Labarta, J.: Thread Fork/Join Techniques for Multi-level Parallelism Exploitation in NUMA Multiprocessors, *Proc. International Conference on Supercomputing* (1999).
7) 岡本雅巳，合田憲人，宮沢 稔，本多弘樹，笠原博徳：OSCARマルチグレインコンパイラにおける階層型マクロデータフロー処理手法，情報処理学会論文誌，Vol.35, No.4, pp.513–521 (1994).
8) 笠原博徳，小幡元樹，石坂一久：共有メモリマルチプロセッサシステム上での粗粒度タスク並列処理，情報処理学会論文誌，Vol.42, No.4, pp.910–920 (2001).
9) 笠原博徳：並列処理技術，コロナ社 (1991).
10) 合田憲人，岩崎 清，岡本雅巳，笠原博徳，成田誠之助：共有メモリ型マルチプロセッサシステム上でのFortran粗粒度タスク並列処理の性能評価，情報処理学会論文誌，Vol.37, No.3, pp.418–429 (1996).
11) Yang, T. and Gerasoulis, A.: DSC: Scheduling parallel tasks on an unbounded number of processors, *IEEE Trans. on Parallel and Distributed Systems*, Vol.5, No.9, pp.951–967 (1994).
12) Kasahara, H. and Yoshida, A.: A Data-Localization Compilation Scheme Using Partial Static Task Assignment, *Journal of Parallel Computing*, Vol.24, No.3, pp.579–596 (1998).

(平成13年8月28日受付)
(平成14年2月13日採録)

吉田 明正（正会員）

1968年生．1991年早稲田大学理工学部電気工学科卒業．1993年同大学大学院修士課程修了．1996年同大学院博士課程修了．博士（工学）．1993年日本学術振興会特別研究員．1995年早稲田大学理工学部電気電子情報工学科助手．1997年東邦大学理学部情報科学科専任講師，現在に至る．並列処理方式，並列化コンパイラ，自動データ分散，スケジューリング等に関する研究に従事．電子情報通信学会，電気学会，IEEE，ACM各会員．