粗粒度タスク並列処理のための階層統合型実行制御手法

吉田明正†

本論文では，階層型マクロタスクグラフを用いた粗粒度タスク並列処理において，異なる階層の粗粒度タスク間並列性を効果的に利用する階層統合型実行制御手法を提案する．従来の粗粒度タスク並列処理では，階層的に定義された粗粒度タスクが，階層的にグルーピングされたプロセッサに割り当てられて実行されていた．このため，対象プログラムの各部分ごとに利用できる並列性の階層が異なる場合，限られたプロセッサ台数では，すべての並列性を利用することが困難であった．この問題点を解決するために，本論文では，階層開始マクロタスクという概念を導入することにより，複数階層の粗粒度タスクを統一的に取り扱い，全プロセッサ上で全階層の並列性を最大限に利用する実行制御手法を提案する．また，4 階層ランダムマクロタスクグラフを用いた性能評価と，SMP 上での OpenMP 実装による SPECfp95 ベンチマークプログラムの性能評価の結果から，提案手法の有効性が確認されている．

Layer-unified Execution Control Scheme for Coarse Grain Task Parallel Processing

Akimasa Yoshida†

This paper proposes a layer-unified execution control scheme to use parallelism among coarse grain tasks across several layers in the coarse grain task parallel processing using hierarchical macrotask graphs. Conventionally, the coarse grain tasks of each layer were executed on hierarchical processor-groups. Therefore, if the parallelism of a target program exists on several different layers, it is difficult to utilize the whole parallelism on the restricted number of processors. To solve such problem, this paper proposes an execution control scheme to deal with the coarse grain tasks of several layers together by using the concept of layer-start macrotasks. In the performance evaluations, the effectiveness of the layer-unified execution control scheme is confirmed by the simulation using 4-layer random macrotask graphs and the implementation using OpenMP on SMP for SPECfp95 benchmark program.

1. はじめに

共有メモリ型マルチプロセッサ用自動並列化コンパイラでは，従来よりループレベル並列化技術[1] が用いられており，たとえば，イリノイ大学とパデュー大学の Polaris[2] やスタンフォード大学の SUIF[3] のような並列化コンパイラでは，データ依存解析技術，プログラムリストラクチャリング技術，データローカリティ最適化技術[4] を組み合わせることによりさまざまな形状のループが効果的に並列化可能になっている．しかしながら，今後さらなる性能向上を目指すためには，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[5)~8)] を利用する粗粒度タスク並列処理が有効と考えられる．

粗粒度タスク並列処理[5] やマルチグレイン並列処理[6] では，粗粒度タスク間の並列性を並列化コンパイラが自動抽出して階層型マクロタスクグラフを生成し，各階層の粗粒度タスクを，グルーピングしたプロセッサに階層的に割り当てて並列処理を行っていた．この場合，対象プログラム中の各階層の粗粒度タスクは，その階層を処理すべきプロセッサグループに割り当てられて実行されるため，十分な台数のプロセッサを確保できない場合には，対象プログラムに内在する全階層の粗粒度タスク間並列性を利用できない可能性がある．

そこで，本論文では，粗粒度タスク並列処理において用いられている階層型マクロタスクグラフ[5),6)] を利用しつつ，対象プログラム中の異なる階層の粗粒度タスクを統一的に取り扱い，異なる階層間にまたがった粗粒度タスク並列性を最大限に利用する階層統合型実行制御手法を提案する．本手法では，階層型マクロタ

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

スクグラフにおいて階層開始マクロタスクを導入し，従来の最早実行可能条件を変換することにより，全階層の粗粒度タスクを統一的に取り扱う実行制御を実現している．

本論文の構成は以下のとおりとする．2 章では粗粒度タスク並列処理の概要を述べる．3 章では提案する階層統合型実行制御の概念を述べ，4 章では階層統合型実行制御を実現するための最早実行開始条件の変換方法，および，OpenMP 等のマルチスレッドによる実装方法について述べる．5 章ではランダムマクロタスクグラフを用いた性能評価，6 章では SPECfp95 の Turb3d ベンチマークプログラムに対して，階層統合型実行制御を実現する並列プログラムを OpenMP により実装し，SMP 上で行った評価について述べる．7 章では関連研究について述べ，8 章でまとめを述べる．

2. 粗粒度タスク並列処理

本章では，粗粒度タスク並列処理の概要を述べる．

2.1 対象マルチプロセッサアーキテクチャ

粗粒度タスク並列処理[5),6)] 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，粗粒度タスク並列処理用の並列化コンパイラにより，OpenMP API を含む並列処理用プログラムを生成し，次にそのプログラムから商用コンパイラで対象マシンの実行コードを生成する方法をとる[5)]．

従来の粗粒度タスク並列処理（本論文中では階層型実行制御と呼ぶ）[5),6)] では，マルチプロセッサシステムの全プロセッサ（PE）を第 0 階層プロセッサグループ（PG）と定義し，第 0 階層 PG 内の PE を複数のグループに分割し，それぞれを第 1 階層 PG と定義する．同様に，第 L 階層 PG 内の PE を複数のグループに分け第 $(L+1)$ 階層 PG を定義する．ただし，最下位階層の PG は 1PE 構成となる．

それに対して，提案する階層統合型実行制御手法を用いた粗粒度タスク並列処理では，プロセッサのグルーピングが不要であり，対象プログラムの全階層の並列性を最大限に利用することが可能となる．

2.2 粗粒度タスク並列処理の実行方式

粗粒度タスク並列処理[5),6)] は，階層的にループやサブルーチン等の粗粒度タスク間の並列性を抽出し，粗粒度タスク（マクロタスク）をプロセッサあるいはプロセッサグループ（PG）に割り当てて並列処理する方式である．

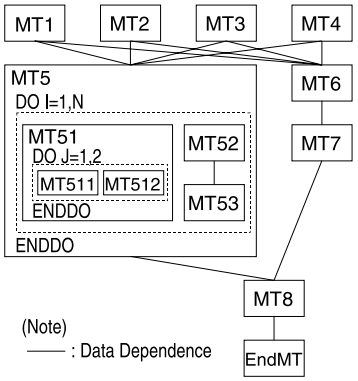

図 1 3 階層プログラムの MTG
Fig. 1 MTG of 3-layer program.

粗粒度タスク並列処理による実行では，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（MT）に分割する．マクロタスクは，擬似代入文ブロック（基本ブロック），繰返しブロック（ループ），あるいは，サブルーチンブロックの 3 種類から構成される[5),6)]．次に，第 1 階層マクロタスク内部に複数のサブマクロタスク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．同様に，第 L 階層マクロタスク内部において，第 $(L+1)$ 階層マクロタスクを定義する．

ここで，繰返しブロックが Doall ループ（あるいはリダクションループ）である場合には，粗粒度並列性を向上させるために，複数の部分 Doall ループに分割し，それらを別々のマクロタスクとして定義する[9)]．この際，従来の階層型実行制御の場合には，当該階層の PG 数あるいはその整数倍に分割し，階層統合型実行制御の場合には，全 PE 数あるいはその整数倍に分割することにより，Doall ループの並列性を有効利用できる．

マクロタスク生成後，各階層のマクロタスク間の制御フローとデータ依存を解析し，階層型マクロフローグラフ[5),6)] を生成する．次に，制御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[5),6)] を解析する．最早実行可能条件は，制御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おり，マクロタスクの実行制御に用いられる．たとえば，**図 1** の MT5 の最早実行可能条件は，**表 1** に示すように 1∧2∧3∧4 と求められ，MT5 は MT1〜MT4 の実行終了後に実行可能となることを表している．

提案する階層統合型実行制御を適用する場合には，

表 1 最早実行可能条件と終了ステートの階層統合型変換
Table 1 Layer-unified conversion for earliest-executable-conditions and finish-states.

MT 番号	最早実行可能条件		終了ステート	
	階層型（従来）	階層統合型	階層型（従来）	階層統合型
MT1	true		1	
MT2	true		2	
MT3	true		3	
MT4	true		4	
MT5 †	1 ∧ 2 ∧ 3 ∧ 4		5	**5S**
MT6	1 ∧ 2 ∧ 3 ∧ 4		6	
MT7	6		7	
MT8	5 ∧ 7		8	
EndMT9	8		9	
MT51 ††	true	**5S**	51	**51S**
MT52	true	**5S**	52	
MT53	52		53	
CtrlMT54	51 ∧ 53		54	
RepMT55	54_{55}		55	
ExitMT56	54_{56}		56	**5**
MT511	true	**51S**	511	
MT512	true	**51S**	512	
CtrlMT513	511 ∧ 512		513	
RepMT514	513_{514}		514	
ExitMT515	513_{515}		515	**51**

（注）† 階層統合型の場合，第 2 階層用の階層開始 MT.
†† 階層統合型の場合，第 3 階層用の階層開始 MT.

従来の最早実行可能条件を解析した後，4 章で述べる方法により階層開始マクロタスクを導入し，各種条件を変換する．なお，各階層のマクロタスクの最早実行可能条件は，階層型マクロタスクグラフ（MTG）[5),6)] によって表すことが可能であり，たとえば，図 1 の MTG は，表 1 の最早実行可能条件を図示したものである．

マクロタスクのスケジューリングに関しては，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．このとき，従来の階層型実行制御では，プロセッサをグルーピングし，階層型マクロタスクグラフ上のマクロタスクを各階層ごとに，対応する階層のプロセッサグループに割り当てる方式をとる[5)]．すなわち，各階層のマクロタスクは当該階層のスケジューラにより割り当てられる．一方，本論文で提案する階層統合型実行制御を用いる場合，後述の階層開始マクロタスクを導入して全階層のマクロタスクを統一的に扱い，それらを実行時にプロセッサに割り当てる方式をとる．

3. 階層統合型実行制御の概念

従来の階層型実行制御をともなう粗粒度タスク並列処理[5),6)] では，各階層の粗粒度タスクは，プロセッサグループ（PG）に階層的に割り当てられて実行される．たとえば，図 1 の 3 階層プログラムでは，第 1 階層が MT1〜MT8 により構成されており，MT5 のループ内部の第 2 階層は，MT51〜MT53 により構成されており，MT51 のループ内部の第 3 階層は MT511〜MT512 により構成されているものとする．

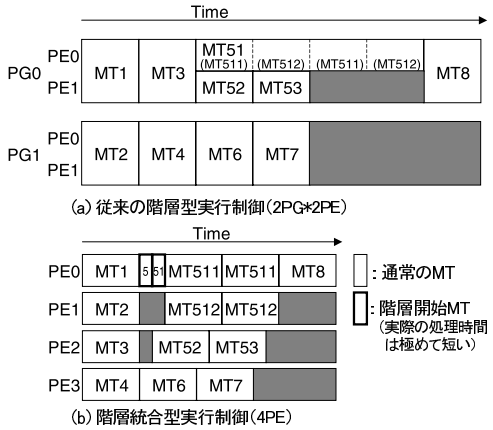

図 2 3 階層プログラム（N = 1）の実行イメージ
Fig. 2 Execution image of 3-layer program (N = 1).

この 3 階層プログラムを従来の階層型実行制御を用いて，2PG（各 PG は 2PE 構成で計 4PE）上で実行したイメージは，**図 2** (a) のようになる．第 1 階層の MT1，MT3，MT5，MT8 は PG0 に割り当てられ，MT2，MT4，MT6，MT7 は PG1 に割り当てられる．次に，MT5 内部の第 2 階層の MT51，MT52，MT53 は，PG0 内の PE0 と PE1 に割り当てられる．ここで，仮に MT1〜MT4，MT6〜MT8 がループ並列処理できない場合，各 MT は 1PG（2PE 構成）に割り当てられているが，PG 内の 1PE 上でシーケンシャル実行されることになり PG（プロセッサグループ）の利用率は低い．また，第 2 階層の MT51 は PG 内の 1PE に割り当てられており，MT51 内の第 3 階層の MT511 と MT512 はシーケンシャルに実行される．

それに対して提案する階層統合型実行制御では，図 2 (b) に示すように，MT1〜MT8 の第 1 階層マクロタスク，MT5 内部の MT51〜MT53 の第 2 階層マクロタスク，MT51 内部の MT511〜MT512 の第 3 階層マクロタスクを統一的に取り扱い，それらを各プロセッサに割り当てて実行することが可能となる．この場合，MT5 はその内部の第 2 階層の開始処理のみを行う階層開始マクロタスク（後述）として扱われ，MT51 もその内部の第 3 階層の開始処理のみを行う階層開始マクロタスクとして扱われる．本手法では，第 1 階層から第 3 階層までの並列性（たとえば MT511，MT512，MT52，MT6 の間の並列性）が同時に利用さ

れており，実行時間が大幅に短縮されることが分かる．

4. 階層統合型実行制御の実現方式

本章では，階層統合型実行制御の実現方式について述べる．

4.1 階層開始マクロタスク

階層統合型実行制御では，全階層のマクロタスクを統一的に取り扱うため，第 L 階層マクロタスクを内部に持つ上位の第 $(L-1)$ 階層マクロタスクを，第 L 階層用の階層開始マクロタスクとして取り扱う．この階層開始マクロタスクは，内部の第 L 階層マクロタスクの実行を開始するために使用される．この階層開始マクロタスクの導入により，当該階層のマクロタスクの実行が可能になったことが保証され，全階層のマクロタスクを同時に取り扱うことが可能となる．

たとえば，図 1 の MT5 の場合，内部に第 2 階層 MT（MT51，MT52，MT53）を含んでおり，MT5 は第 2 階層用の階層開始マクロタスクとして扱われる．同様に，図 1 の MT51 の場合，内部に第 3 階層 MT（MT511，MT512）を含んでおり，MT51 は第 3 階層用の階層開始マクロタスクとして扱われる．

4.2 最早実行可能条件の階層統合型変換

階層開始マクロタスクの導入により，従来の階層ごとに求めた最早実行可能条件[5),6)] を階層統合型実行制御用に変換する．具体的には，第 L 階層マクロタスクの最早実行可能条件が「true」（すなわちその階層が実行可能になればすぐに実行可能）である場合，その条件を「第 L 階層用の階層開始マクロタスク MT_i の終了」に置き換える．ここで，階層開始マクロタスクとしての MT_i の終了ステート（MT 終了時のスケジューラへの通知信号）は iS と表記し，本来の MT_i の内部全体の終了ステート[5),6)] は従来どおり i と表記する．

たとえば，図 1 の各 MT の最早実行可能条件[5),6)] は表 1 に示すとおりである．第 2 階層の MT51 と MT52 の最早実行可能条件は，従来の階層型実行制御用では「true」であるため，階層統合型実行制御用では「第 2 階層用の階層開始マクロタスク MT5 の終了（5S）」と置き換える．同様に，第 3 階層の MT511 と MT512 の最早実行可能条件は，階層型実行制御用では「true」であるため，階層統合型実行制御用では「第 3 階層用の階層開始マクロタスク MT51 の終了（51S）」と置き換える．この結果，MT51 は，5S（階層開始 MT としての MT5 の終了）の通知信号を検出した後に実行可能となり，同様に，MT511 は，51S（階層開始 MT としての MT51 の終了）の通知信号を検出した後に実行可能となる．

なお，最早実行可能条件において，i は MT_i の終了，$(i)_j$ は MT_i から MT_j への分岐，i_j は MT_i から MT_j への分岐と MT_i の終了を表している．また，EndMT（終了処理），CtrlMT（当該階層の繰返し判定処理），RepMT（当該階層の繰返し更新処理），ExitMT（当該階層の終了処理）は制御に用いられるダミーマクロタスクである．

4.3 終了ステートの階層統合型変換

粗粒度タスク並列処理では，各 MT の実行終了時に，その MT の終了ステートを，ダイナミックスケジューリング用のステート管理テーブルに記録する方式をとる．ダイナミックスケジューリングの際には，このステート管理テーブルを用いることにより，新たに実行可能な MT を検出することが可能となる．

ここで，階層統合型実行制御を実現する場合，第 L 階層マクロタスクを内部に持つ第 $(L-1)$ 階層マクロタスク MT_i を，階層開始マクロタスクとして扱うため，階層開始マクロタスクとしての MT_i の実行終了と，MT_i 内部の第 L 階層の実行終了は別々に取り扱う必要がある．そこで，本方式では，階層開始マクロタスクとしての MT_i の実行終了を表す終了ステート iS は，階層開始マクロタスク自身に発行させ，MT_i 内部の第 L 階層の実行終了を表す終了ステート i は，第 L 階層の ExitMT に発行させている．

たとえば，図 1 の各マクロタスクの終了ステートは表 1 に示すとおりであり，階層開始マクロタスク MT5 の終了ステートは 5S であり，本来の MT5 の終了ステート 5（MT5 内部の階層の終了を意味する）は，その内部の ExitMT56（図 1 では省略）により発行される．同様に，階層開始マクロタスク MT51 の終了ステートは 51S であり，本来の MT51 の終了ステートである 51 は，その内部の ExitMT515（図 1 では省略）により発行される．

4.4 階層統合型レディマクロタスクキュー

粗粒度タスク並列処理においてダイナミックスケジューリングを適用する場合，各マクロタスクはその最早実行可能条件[5),6)] が満たされた後，レディマクロタスクキューに投入され，プライオリティの高い（すなわち，Critical-Path（CP）長の大きい）マクロタスクから順にレディマクロタスクキューから取り出されてプロセッサに割り当てられる．

ここで，従来の階層型実行制御の場合，マクロタスク内のサブマクロタスクのダイナミックスケジューリングは，そのマクロタスク単位で独立に行えばよいため，マクロタスクごとに用意したレディサブマクロタ

スクキューを使用していた．

それに対して，階層統合型実行制御を実現する場合，異なる階層のマクロタスクを統一的に扱うため，全階層のマクロタスクを対象とした階層統合型レディマクロタスクキューを導入する．すなわち，各階層のマクロタスクは，その最早実行可能条件が満たされた後，階層統合型レディマクロタスクキューに投入される．

4.5 階層統合型ダイナミックスケジューリング

階層統合型実行制御をともなうダイナミックスケジューリングは，分散型ダイナミックスケジューリングあるいは集中型ダイナミックスケジューリングにより実現が可能である．分散型ダイナミックスケジューリングの場合は，各 PE がスケジューリング処理とマクロタスク処理の両方を行う方式であり，集中型ダイナミックスケジューリングの場合は，1PE がスケジューリング処理を専属で行い，それ以外の PE がマクロタスク処理を行う方式である．

以下にスケジューリング処理の手順を示す．

(i) 全階層のマクロタスクの中から最早実行可能条件を満たすマクロタスクを，階層統合型レディマクロタスクキューに投入する．

(ii-a) 分散型ダイナミックスケジューリング時は，階層統合型レディマクロタスクキューから CP 長の大きいマクロタスクを取り出し，自 PE に割り当てる．

(ii-b) 集中型ダイナミックスケジューリング時は，階層統合型レディマクロタスクキューから CP 長の大きいマクロタスクを取り出し，アイドル PE に割り当てる．

(iii) 分散型ダイナミックスケジューリング時は，自 PE に割り当てられたマクロタスクの処理を行う．

(iv) EndMT が終了していない間は (i) に戻る．

なお，CP 長の値としては，全階層のマクロタスクの CP 長を絶対的に比較するため，最上位階層 MTG の出口ノードからの CP 長（絶対 CP 長）[10] を用いる．

4.6 マルチスレッドによる実装方法

階層統合型実行制御をともなう粗粒度タスク並列処理用コードを，OpenMP や POSIX スレッド等のマルチスレッドを用いて実現する場合，前述のように分散型ダイナミックスケジューリング方式あるいは集中型ダイナミックスケジューリング方式による実装が可能である[5),11)]．ここでは，6.2 節の性能評価で採用した OpenMP による分散型ダイナミックスケジューリング方式の実現方法について述べる．

階層統合型実行制御を実現するためには，スケジューリング処理部とマクロタスク処理部から構成されるスレッドコードを PE 台数分生成する．OpenMP による実装では，`$!OMP PARALLEL`，`$!OMP SECTIONS`，`$!OMP SECTION` により，各スレッドコードを記述する．これらのスレッドは実行開始時に各 PE 上において 1 回のみ生成しており，スレッド生成にともなうオーバヘッドを軽減している．

各スレッドコード（`$!OMP SECTION` 部分）では，各 PE 上でマクロタスクの処理を終えるたびに，スケジューリング処理部でスケジューリングを行い，自 PE に新たに割り当てられたマクロタスクの処理を行う．なお，階層統合型レディマクロタスクキューのアクセスに対しては排他制御を行う．OpenMP 実装の場合には，`$!OMP CRITICAL` により排他制御を行っている．

5. ランダムマクロタスクグラフを用いた性能評価

本章では，提案する階層統合型実行制御をともなう粗粒度タスク並列処理を，従来の階層型実行制御と比較しながら，4 階層ランダムマクロタスクグラフを用いてシミュレーションにより性能評価する．

5.1 4 階層ランダムマクロタスクグラフ

本シュミレーションでは，**表 2** に示す各カテゴリ（SSSS〜LLLL）ごとに，20 個の 4 階層マクロタスクグラフ（MTG）を用意し，計 220 個の 4 階層 MTG を用いて性能評価を行う．各カテゴリごとに各階層の MT 間並列性は異なっており，カテゴリを示す記号は，第 1 階層から第 4 階層までの並列性の大小を表しており，S は並列性が小さく，L は並列性が大きいことを表している．4 階層 MTG は，各階層に 1 つ以上の

表 2 4 階層ランダム MTG の分類
Table 2 Categorization of 4-layer random MTGs.

MTG カテゴリ	MT 間並列性				4 階層 MTG の構成	
	第 1 階層	第 2 階層	第 3 階層	第 4 階層	MTG 数 †	MT 数 †
SSSS	小	小	小	小	4	38
SSSL	小	小	小	大	4	62
SSLS	小	小	大	小	5	83
SLSS	小	大	小	小	7	100
LSSS	大	小	小	小	9	116
SSLL	小	小	大	大	5	152
SLLS	小	大	大	小	13	222
LLSS	大	大	小	小	20	292
SLLL	小	大	大	大	13	417
LLLS	大	大	大	小	36	647
LLLL	大	大	大	大	36	1247

（注）† 各 MTG カテゴリにおける平均値．

表 3 4 階層ランダム MTG の 16PE 上での速度向上率
Table 3 Speedup ratio of 4-layer random MTGs on 16 PEs.

MTG カテゴリ	階層統合型実行制御		従来の階層型実行制御（各 PG 構成）における速度向上率										
	速度向上率	階層型の最善 PG 比	最善 PG	1*1*1*16	1*1*16*1	1*16*1*1	16*1*1*1	1*1*4*4	1*4*4*1	4*4*1*1	1*2*2*4	4*2*2*1	2*2*2*2
SSSS	2.42	9%	2.21	1.28	1.23	1.14	1.07	1.69	1.46	1.24	2.02	1.59	2.17
SSSL	4.30	34%	3.21	2.61	1.08	1.05	1.02	2.79	1.15	1.08	3.20	1.18	2.13
SSLS	4.16	25%	3.34	1.36	2.18	1.04	1.02	3.28	2.34	1.06	2.61	1.95	2.62
SLSS	4.02	28%	3.13	1.33	1.20	1.82	1.01	1.73	2.27	1.86	3.08	2.11	2.98
LSSS	3.87	20%	3.17	1.28	1.20	1.12	1.64	1.64	1.39	1.84	1.94	2.40	3.17
SSLL	7.41	46%	5.05	2.73	1.70	1.01	1.00	5.01	1.75	1.02	4.57	1.66	2.99
SLLS	6.29	61%	3.89	1.33	2.00	1.59	1.00	3.09	3.39	1.61	3.80	2.84	3.74
LLSS	5.39	34%	3.90	1.27	1.21	1.69	1.48	1.64	2.14	2.57	2.78	2.99	3.81
SLLL	9.59	57%	6.04	2.75	1.64	1.33	1.00	4.84	2.25	1.34	6.01	2.05	3.74
LLLS	9.18	71%	5.32	1.32	2.03	1.48	1.59	3.07	3.23	2.34	3.60	4.25	5.27
LLLL	13.50	105%	6.62	2.76	1.62	1.45	1.47	4.83	2.35	2.12	6.33	3.19	5.85

（注）PG 構成 n_1*n_2*n_3*n_4 において，n_i は第 i 階層 PG（プロセッサグループ）の数を表す．

MTG を持っており，そのグラフを構成する MTG 数と MT 数は，カテゴリごとの平均で表 2 の MTG 数と MT 数のとおりである．

各階層の MTG 生成は，文献 12) で行われているように，MTG 段数を決定し，各段において MTG 幅の MT を配置し，異なる段の 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，階層生成率（MT が内部にサブ MT を持つ割合）を 0.1，内部の階層を実行するループ回転数を 1〜2，MTG 段数を 4，MTG 幅を 1〜3（並列性小）または 7〜9（並列性大），MT の先行データ依存エッジ数を 1〜3（並列性小）または 7〜9（並列性大），MT の処理時間を 10〜100 [u.t.] としている．

5.2 シミュレーションによる性能評価

本節では，階層統合型実行制御と従来の階層型実行制御をともなう粗粒度タスク並列処理を，5.1 節で生成された 4 階層 MTG に対して適用し，16PE 上でシミュレーション実行する．本シミュレーションでは，プログラム中の並列性利用を比較するためのものであり，ダイナミックスケジューリングによるオーバヘッドは考慮していない．

シミュレーション結果は**表 3** のとおりであり，各カテゴリに属する 20 個の 4 階層 MTG の 1PE に対する速度向上率を求め，その平均値を示している．提案する階層統合型実行制御では，16PE（プロセッサのグルーピングは不要）の結果を示している．従来の階層型実行制御では，各カテゴリにおける各階層の並列性の大小を考慮して，16PE で有効と考えられる 10 通りの PG（プロセッサグループ）構成，1*1*1*16，1*1*16*1，1*16*1*1，16*1*1*1，1*1*4*4，1*4*4*1，4*4*1*1，1*2*2*4，4*2*2*1，2*2*2*2 により評価している．なお，各 MTG ごとに最善の PG 構成を適用した結果を最善 PG の欄に示している．

たとえば，1*1*1*16 の場合，第 1 階層 PG 数=1，第 2 階層 PG 数=1，第 3 階層 PG 数=1，第 4 階層 PG 数=16 であるため，第 1 階層から第 3 階層までの並列性はまったく利用できず，第 4 階層において 16 並列の実行が可能となる．2*2*2*2 では，第 1 階層から第 4 階層までの各階層において 2 並列の実行が可能となる．

カテゴリ別にみると，カテゴリ SSLL は第 3・第 4 階層の並列性が大きい MTG の分類であり，従来の階層型実行制御では，第 3・第 4 階層に多くのプロセッサを割り当てた 1*1*4*4 の PG 構成（計 16PE）のときが，このカテゴリの 20 例の平均で最も性能が良く，1PE の 5.01 倍を達成している．また，各 4 階層 MTG ごとに最適な PG 構成を採用した場合には，1PE の 5.05 倍の速度向上が得られている．それに対して，提案する階層統合型実行制御では 16PE 上で 7.41 倍の速度向上が得られており，階層型実行制御と比べて 46%の性能向上が得られていることが分かる．

また，いずれのカテゴリにおいても，階層統合型実行制御手法は，従来の階層型実行制御手法（最善 PG 構成の場合）と比べて，9〜105%の性能向上を達成可能であり，本手法の有効性が確認できる．これは，階層統合型実行制御では，第 1 階層から第 4 階層までのすべての階層において，全プロセッサを使用して並列実行することが可能となり，全階層の並列性が有効に利用できたことを意味している．

6. SPECfp95 ベンチマークプログラムを用いた SMP 上での性能評価

本章では，提案する階層統合型実行制御をともなう粗粒度タスク並列処理を，SPECfp95 ベンチマークプ

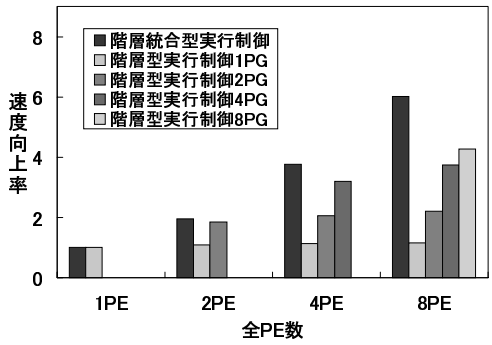

図 3 Turb3d プログラムのシミュレーションによる実行結果
Fig. 3 Execution result of Turb3d program by simulation.

ログラム Turb3d を用いて性能評価する．

6.1 Turb3d プログラムを用いた階層統合型実行制御と階層型実行制御の性能比較

SPECfp95 の Turb3d プログラムは，2,101 行の Fortran ソースコードから構成されており，Navier-Stokes 方程式を解くことにより乱気流シミュレーションを行うことが可能である．このプログラムのメインループの MTG は，ループディストリビューション適用後 62 個のマクロタスク（MT）から構成されている．この各 MT 内部にはサブマクロタスク（サブ MT）が含まれており，そのサブ MT が Doall 処理あるいはリダクション処理可能な Do ループの場合には，PE 数あるいはその整数倍（本評価では 8）に分割しておく[9]．

本評価では，まず，Turb3d プログラムを Sun Ultra80 上でシーケンシャル実行することにより，各 MT（またはサブ MT）の実行時間と条件分岐のプロファイルを取得する．なお，62 個の MT から構成されるメインループは 12 回転実行されるため，それぞれに対してプロファイル情報を取得しておく．

次に，上述の実行プロファイルと MTG 情報（各 MT の最早実行可能条件）を入力ファイルとし，5.2 節で用いたシミュレーション環境により，階層統合型実行制御と従来の階層型実行制御における並列処理時間を測定すると，**図 3** の結果が得られた．図 3 において，従来の階層型実行制御の場合には，プロセッサのグルーピングが必要であり，1PG 構成，2PG 構成，4PG 構成，8PG 構成と変化させて並列実行させている．

図 3 の結果から，いずれの PE 数においても，提案する階層統合型実行制御手法は，従来の階層型実行制御手法（すべての PG 構成）と比べて，顕著に処理時間が短縮されている．たとえば，8PE 上での従来の階層型実行制御の場合，1PG 構成（1PG*8PE），2PG 構成（2PG*4PE），4PG 構成（4PG*2PE），8PG 構成（8PG*1PE）の順に速度向上率が上がっており，8PG 構成のときに 4.27 倍の速度向上率となっている．それに対して，階層統合型実行制御の場合には，6.02 倍の速度向上率を達成しており，従来の階層型実行制御 8PG 構成と比べると，処理時間が 30%短縮されている．

それゆえ，階層統合型実行制御手法は，ベンチマークプログラム Turb3d の場合，与えられたプロセッサ上で全階層の並列性を最大限に利用することが可能であり，いずれの PE 数においても，従来の階層型実行制御手法より高い性能を達成できることが確認された．

6.2 SMP 上での OpenMP 実装による Turb3d プログラムの性能評価

本節では，階層統合型実行制御をともなう粗粒度タスク並列処理を OpenMP API を用いて実装し，Turb3d ベンチマークプログラムに対して，SMP（Sun Ultra80）上で性能評価した結果について述べる．

Sun Ultra80 は，UltraSPARC II（450 MHz）プロセッサ 4 台を，Gigaplane インターコネクトにより，1 GB の集中共有メモリに接続した SMP マシンである．OS は Solaris 8 であり，Fortran コンパイラは Sun Forte Compilers Version 6 update 1 の f95 を使用している．

1PE 上でのシーケンシャル実行の場合には，オリジナルのベンチマークプログラムに対して，`-fast` のコンパイルオプションを適用して実行しており，その実行結果は**表 4** に示すとおりである．

次に，提案する階層統合型実行制御をともなう粗粒度タスク並列処理の場合には，4.6 節で述べたように，オリジナルの Turb3d プログラム（2,101 行）に対して，ダイナミックスケジューリング処理部とマクロタスク処理部からなるスレッドコードを追加し，OpenMP API を用いた並列プログラムを作成した．この階層統合型実行制御の並列プログラムに対して，`-fast -openmp -stackvar` のオプションでコンパイルし，Ultra80 上で並列実行した結果は，表 4 に示すとおりである．この場合，4PE 上での並列処理時間は，シーケンシャル実行時間の 1/2.90 に短縮されている．

このときの実行トレースは**図 4** のようになっており，Turb3d プログラムのメインループ（62 個の MT とそのサブ MT から構成されており，Doall あるいはリダクション処理可能な Do ループは 4 分割済み）が 12 回転分実行されている．本トレースでは，4PE 上で計 1,418 個の MT（またはサブ MT）が並列実行さ

表 4 Sun Ultra80 上での Turb3d プログラムの実行結果
Table 4 Execution result of Turb3d program on Sun Ultra80.

実行方式	PE 数	実行時間 [s]	速度向上率	各 PE の実行 MT 数†	各 PE の MT 処理時間合計 [s]†	PE 利用率 [%]†
シーケンシャル実行	1	21.950	1.00			
階層統合型実行制御	1	21.983	0.99	1418	21.973	99.9
階層統合型実行制御	2	12.148	1.81	709	12.071	99.4
階層統合型実行制御	3	9.008	2.44	473	8.694	96.5
階層統合型実行制御	4	7.558	2.90	355	7.401	97.9

（注）† 全 PE の平均値.

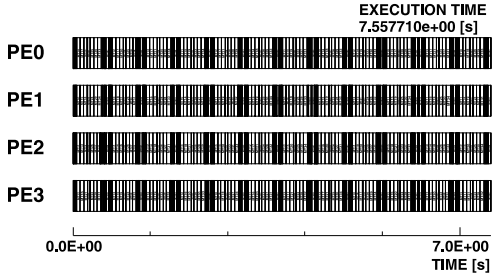

図 4 Ultra80 上での階層統合型実行制御による Turb3d の実行トレース
Fig. 4 Execution trace of Turb3d by layer-unified execution control on Ultra80.

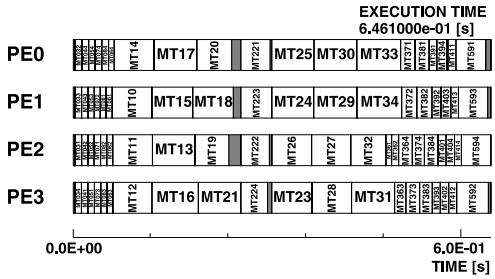

図 5 Ultra80 上での階層統合型実行制御による Turb3d の実行トレース（1 回転目メインループ）
Fig. 5 Execution trace of Turb3d (main loop at first time) by layer-unified execution control on Ultra80.

れており，提案手法は異なる階層の MT 間並列性を最大限に利用しており，PE 利用率が高いことが分かる．この実行トレースの各 PE において黒くなっている部分は，ほとんどが処理時間の小さいマクロタスクの集合である．ここで，図 4 の実行トレースにおける最初の部分，すなわち Turb3d プログラムのメインループの第 1 回転目の実行部分のみを拡大表示したものが，**図 5** の実行トレースでありダイナミックスケジューリングオーバヘッドがきわめて小さいことが分かる．

このことは，表 4 の 4PE 実行においても，PE 利用率（各 PE で実行した MT の処理時間合計/全体の並列処理時間）の 4PE 平均値が 97.9%に達しており，MT 処理時間以外のダイナミックスケジューリング時間およびアイドル時間がきわめて小さいことが分かる．なお，表 4 の Ultra80 上での並列実行では，PE 数の増加にともない，全 MT の処理時間合計（各 PE の MT 処理時間合計の平均値*PE 数）が増加しているが，今後，キャッシュ有効利用を考慮したデータローカリティ最適化技術を開発することにより，集中共有メモリへのアクセスが軽減されれば，全 MT の処理時間合計が短縮可能となり，さらなる性能向上が期待できる．

以上の結果，提案する階層統合型実行制御手法は OpenMP API により実装したところ，並列性を最大限に利用しており，かつ，ダイナミックスケジューリングオーバヘッドもきわめて小さいことが分かる．それゆえ，提案手法は OpenMP を用いたマルチスレッドコードにより，低オーバヘッドで実装可能であることが確認された．

7. 関 連 研 究

粗粒度タスク並列処理において，階層型マクロタスクグラフの並列性を考慮し，適切なプロセッサのグルーピング（クラスタリング）を求める方法が文献 13) で提案されているが，マクロタスクの階層型実行制御を前提としているため，限られたプロセッサ数の環境下では，全階層の並列性利用に限界がある．

また，粗粒度タスク並列処理においてプロセッサグループ（プロセッサクラスタ）の利用率を向上させるために，複数のマクロタスクを同一プロセッサグループに多重に割り当てる実行手法が文献 10) で提案されているが，異なるマクロタスク内のサブマクロタスク間並列性を最大限に利用するものであり，異なる階層のマクロタスク間並列性を有効利用することはできない．

実行開始条件を用いた粗粒度タスク間並列性検出をループに拡張する方法が文献 14) で提案されているが，3 階層以上のプログラムへの適用について言及されておらず，また，不定回ループでは実行中に実行開始条件を生成・登録しなければならないため，実アプ

リケーションへの適用は難しいものと思われる．

8. おわりに

本論文では，粗粒度タスク並列処理のための階層統合型実行制御手法を提案した．本手法では，粗粒度タスク並列処理の行われる全階層の粗粒度タスクを統一的に扱い，それらをプロセッサに割り当てることにより，全階層にまたがった粗粒度タスク間並列性を最大限に利用することが可能である．

階層統合型実行制御手法を 4 階層ランダムマクロタスクグラフを用いてシミュレーションにより評価したところ，各階層の並列性の大小にかかわらず，処理時間が従来の階層型実行制御手法より顕著に短縮されており，階層統合型実行制御手法の有効性が確認できた．

SPECfp95 ベンチマークプログラム Turb3d を用いた性能評価の場合，階層統合型実行制御手法は，8PE のシミュレーション実行において，従来の階層型実行制御手法より処理時間が 30%短縮可能であり，その有効性が確かめられた．さらに，階層統合型実行制御をともなう並列処理用コードを OpenMP を用いて実装し，Sun Ultra80 上で実行した結果，階層統合型実行制御手法は，4PE 実行において PE 利用率が 97.9%に達しており，全階層の並列性を最大限に利用しつつ，低スケジューリングオーバヘッドで実装可能であることが確認された．

今後の課題としては，階層統合型実行制御と階層型実行制御を組み合わせたハイブリッド実行制御手法の開発や，階層統合型実行制御においてデータローカリティ最適化技術[9] を導入し，キャッシュ有効利用を実現することがあげられる．

参考文献

1) Wolfe, M.: *High performance compilers for parallel computing*, Addison-Wesley Publishing Company (1996).
2) 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).
3) Hall, M.W., Anderson, J.M., Amarasinghe, S.P., et al.: Maximizing multiprocessor performance with the SUIF compiler, *IEEE Computer* (1996).
4) Lim, A.W. and Lam, M.S.: Cache optimizations with affine partitioning, *Proc. 10th SIAM Conference on Parallel Processing for Scientific Computing* (2001).
5) 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 (2001).
6) 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).
7) Martorell, X., Ayguade, E., Navarro, N., et al.: Thread fork/join techniques for multi-level parallelism exploitation in NUMA multiprocessors, *Proc. International Conference on Supercomputing* (1999).
8) Brownhill, C.J., Nicolau, A., Novack, S. and Polychronopoulos, C.D.: Achieving multi-level parallelization, *Proc. ISHPC'97* (Nov. 1997).
9) Kasahara, H. and Yoshida, A.: A data-localization compilation scheme using partial static task assignment, *Journal of Parallel Computing*, Vol.24, No.3 (1998).
10) 吉田明正：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，情報処理学会論文誌，Vol.43, No.4 (2002).
11) 吉田明正，荒牧智子，大森友寛：粗粒度タスク並列処理における階層統合型スケジューリング，情報処理学会研究報告，No.2002-ARC-149-21 (2002).
12) Yang, T. and Gerasoulis, A.: DSC: Scheduling parallel tasks on an unbounded number of processors, *IEEE Trans. Parallel and Distributed Systems*, Vol.5, No.9 (1994).
13) 小幡元樹，白子　準，神長浩気，石坂一久，笠原博徳：マルチグレイン並列処理のための階層的並列性制御手法，情報処理学会論文誌，Vol.44, No.4 (2003).
14) 本多弘樹，合田憲人，岡本雅巳，笠原博徳：実行開始条件による並列性検出手法―ループへの拡張，情報処理学会並列処理シンポジウム (1993).

(平成 16 年 6 月 23 日受付)
(平成 16 年 10 月 4 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．2004 年同大学理学部情報科学科助教授，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．