社団法人 情報処理学会　研究報告
IPSJ SIG Technical Report

2003－ARC－154　(13)
2003／8／5

階層型マクロタスクグラフのための異階層タスクの統合実行制御手法

吉　田　明　正†

本稿では，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方式を提案する．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，階層的にクラスタリ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．しかしながら，対象プログラムの持つ各階層の並列性によってはプロセッサを有効利用できない可能性がある．この問題点に対処するために，本稿では，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，全プロセッサをクラスタリングすること無しに最大限に利用する実行制御手法を提案する．また，本手法を SMP 上で OpenMP を用いて実装し，SPECfp95 の Turb3d ベンチマークプログラムを用いて評価したところ，本手法の有効性が確認された．

Unified Execution Control Scheme of Different-Layer Tasks for Hierarchical Macrotask Graph

Akimasa Yoshida†

This paper proposes a layer-unified execution control scheme to utilize parallelism among coarse grain tasks across several layers in coarse grain task parallel processing using hierarchical macrotask graphs. Conventionally, coarse grain tasks of each layer were executed on hierarchical processor clusters. However, several programs may decrease processor utilization factor owing to lack of parallelism among coarse grain tasks on each layer. Therefore, this paper proposes an execution control scheme to deal with coarse grain tasks of several layers together by using concept of layer-start macrotasks. Also, according to the performance evaluations using SPECfp95 Turb3d benchmark program on SMP, effectiveness of the proposed scheme was confirmed.

1. はじめに

近年，共有メモリ型マルチプロセッサシステム上での並列化コンパイラを用いた並列処理ではループ並列化手法[1] が広く用いられており，例えば，イリノイ大学の Polaris[2] やスタンフォード大学の SUIF[3] 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，ループリストラクチャリング手法，データローカリティ最適化手法[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，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)~7)] 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),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プロセッサクラスタ（プロセッサグループ）に階層的に割り当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クラスタに割り当てられて実行されるため，プロセッサ数に制限がある場合には，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．また，粗粒度タスク並列処理においてプロセッサクラスタ利用率を向上させるために，複数の粗粒度タスクを同一プロセッサクラスタに多重に割り当てる実行手法[8)] が提案されているが，異なる粗粒度タスク内のサブ粗粒度タスク間並列性を最大限に利用するものであり，異なる階層の粗粒度タスク間並列性を有効利用することはできない．一方，実行開始条件による粗粒度タスク間並列性抽出をループに拡張する方法[9)] も提案されているが，実行開始条件が複雑になり実マシン上では評価されていない．

そこで，本稿では，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を利用し，対象プログラム中の異なる階層の粗粒度タスクを統一

† 東邦大学 理学部 情報科学科

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，階層型マクロタスクグラフにおいて階層開始マクロタスク[10]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．

本稿の構成は以下の通りとする．第 2 章では粗粒度タスク並列処理の概要を述べる．第 3 章では提案する階層統合型実行制御手法の概念を述べ，第 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 のようなマルチスレッドによる実装方法について述べる．第 5 章ではシミュレーションによる従来手法との性能比較，第 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．

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．例えば，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，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[6)]．

従来の階層型実行制御方式による粗粒度タスク並列処理では，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 L 階層 PC 内の PE を複数のグループに分け第 $L+1$ 階層 PC を定義する．それに対して，提案する階層統合型実行制御方式を用いた粗粒度タスク並列処理では，プロセッサのクラスタリングが不要であり，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クラスタ（PC）に割り当て並列処理する方式である．

粗粒度タスク並列処理手法では，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（基本ブロック），繰り返しブロック（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．

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，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おり，マクロタスクの実行制御に用いられる．

提案する階層統合型実行制御方式を適用する場合には，従来の最早実行可能条件を解析した後，第 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，各種条件の変換を行う．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，例えば，**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．

また，粗粒度タスク並列処理では，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方式では，プロセッサをクラスタリングし，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クラスタに割り当てる方式をとる．一方，本論文で提案する階層統合型実行制御方式を用いる場合，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（クラスタリングは不要）に割り当てる方式をとる．

3. 階層統合型実行制御方式の概念

従来の階層型実行制御方式を伴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，プロセッサクラスタ（PC）に階層的に割り当てられて実行される．例えば，図 1 の 3 階層プログラム（最内側階層の各 MT の処理時間は同一であり，N=1 とする）を 2PC（各 PC 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 PC0 に割り当てられ，MT2，MT4，MT6，MT7 は PC1 に割り当てられる．次に，MT5 内部の第 2 階層の MT51，MT52，MT53 は，PC0 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 1PC（2PE 構成）に割り当てら

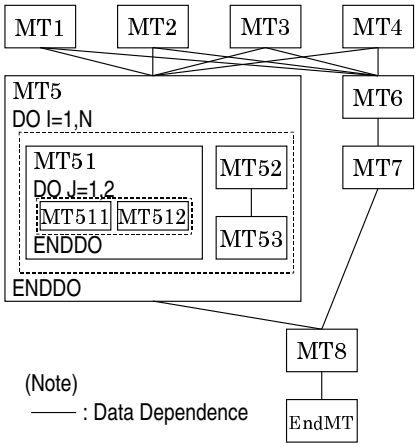

図 1 3 階層プログラムの MTG

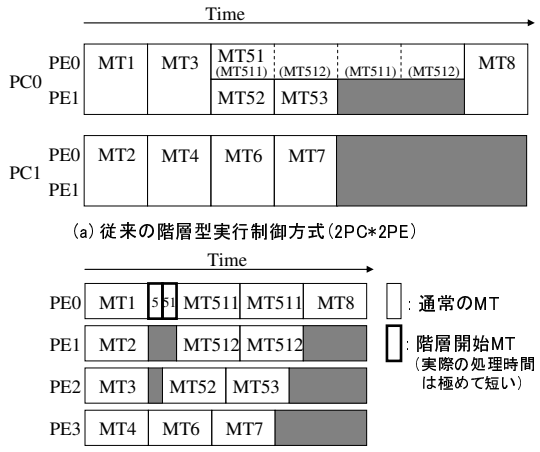

図 2 3 階層プログラム (N=1) の実行イメージ

れているが，PC 内の 1PE 上でシーケンシャル実行されることになりプロセッサクラスタ利用率は低い．また，第 2 階層の MT51 は PC 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方式では，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．本方式では，第 1 階層から第 3 階層までの並列性（例えば 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わかる．

4. 階層統合型実行制御方式の実現方式

本章では，階層統合型実行制御方式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方式では，全階層のマクロタスクを統一的に取り扱うため，内部に第 L 階層マクロタスクを持つ第 $L-1$ 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[10] 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．

例えば，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，従来の階層毎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更する．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」（即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，スケジューリングに用いられる階層開始マクロタスク MT_i の終了ステートは i_s と表記し，MT_i の内部全体の終了ステート[5),6)] は従来通り i と表記する．

例えば，図 1 の各 MT の最早実行可能条件[5),6)] は表 1 に示す通りである．第 2 階層の MT51 と MT52 の最早実行可能条件は従来の階層型用では「true」であるため，「その階層の階層開始マクロタスク MT5 の終了（5_s）」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用では「true」であるため，「その階層の階層開始マクロタスク MT51 の終了（51_s）」と置き換える．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（当該階層の繰り返し判定処理），RepMT（当該階層の繰り返し更新処理），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 終了ステート発行の階層統合型変換

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

表 1 最早実行可能条件と終了ステート発行の階層統合型変換

MT 番号	最早実行可能条件		終了ステート発行	
	階層型	階層統合型	階層型	階層統合型
MT1	true		1	
MT2	true		2	
MT3	true		3	
MT4	true		4	
MT5 †	1 ∧ 2 ∧ 3 ∧ 4		5	**5_s**
MT6	1 ∧ 2 ∧ 3 ∧ 4		6	
MT7	6		7	
MT8	5 ∧ 7		8	
EndMT9	8		9	
MT51 ††	true	**5_s**	51	**51_s**
MT52	true	**5_s**	52	
MT53	52		53	
CtrlMT54	51 ∧ 53		54	
RepMT55	54_{55}		55	
ExitMT56	54_{56}		56	**5**
MT511	true	**51_s**	511	
MT512	true	**51_s**	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.

式をとる[5]．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，階層統合型スケジューリングを実現する場合，内部に第 L 階層マクロタスクを持つ第 $L-1$ 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 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 MT_i の終了ステート i_s は，階層開始マクロタスク自身に発行させ，内部の第 L 階層の終了ステート i は，第 L 階層の ExitMT に発行させる方法を採用する．

例えば，図 1 の各マクロタスクの終了ステートは表 1 に示す通りであり，階層開始マクロタスク MT5 の終了ステートは 5_s 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 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 51_s 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 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（MT）はその最早実行可能条件[5),6)] 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 から順にレディMT キューから取り出されて PE に割り当てられる．ここで，従来の階層型実行制御方式の場合，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 単位で独立に行なえばよいため，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，階層統合型実行制御方式を実現する場合，異なる階層の MT を統一的に扱うため，全階層の MT を対象とした階層統合型レディMT キューを導入する．即ち，各階層の MT 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MT 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

階層統合型ダイナミックスケジューリングでは，階層統合型レディMT キューを CP 長（Critical Path 長）順にソートし，キューの先頭からレディマクロタスクを取り出し，アイドルプロセッサに割り当てる．ここで，CP 長の値としては，全階層の MT の CP 長を絶対的に比較するため，絶対 CP 長[8)] を用いる．以下に具体的な手順を示す．

(i) 全階層の MT の中から最早実行可能条件を満たす MT を，階層統合型レディMT キューに投入する．
(ii) 階層統合型レディMT キューを，MT の CP 長順にソートする．
(iii) 階層統合型レディMT キューから先頭（CP 長が最大）の MT を取り出し，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

粗粒度タスク並列処理用プログラムを，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，分散型ダイナミックスケジューリングと集中型ダイナミックスケジューリングの実装が可能である[6)]．

提案する階層統合型実行制御方式においても，どちらの実装も可能であるが，本稿では，プロセッサ数が十分にない場合を想定して，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を実現する方法について述べる．

階層統合型実行制御方式を実現するためには，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 1 回のみ生成しておりオーバヘッドを軽減している．

各スレッドコード（`$!OMP SECTION` 部分）では，PE 上でマクロタスクの実行を終える度に，ダイナミックスケジューリング処理部を呼び出してスケジューリングを行い，新たに割り当てられたマクロタスクを実行する．なお，階層統合型レディMT キューのアクセスに対しては排他制御を行う．OpenMP 実装の場合には，`$!OMP CRITICAL` により排他制御を行う．

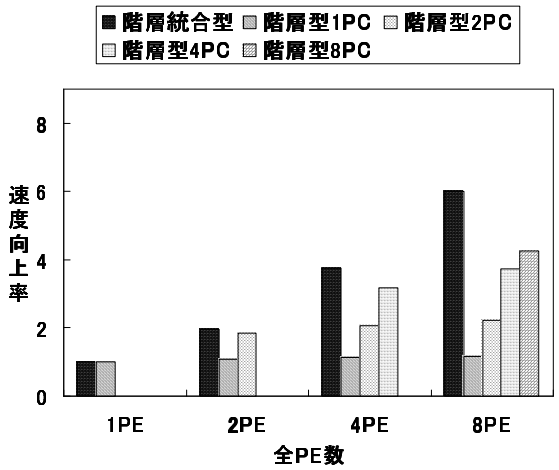

図 3 Turb3d プログラムのシミュレーションによる実行結果
Fig. 3 Execution result of Turb3d program by simulation

5. 階層統合型実行制御方式のシミュレーションによる性能評価

本章では，提案する階層統合型実行制御を用いた粗粒度タスク並列処理を，SPECfp95 ベンチマークプログラム Turb3d を用いて性能評価する．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，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 個のマクロタスクから構成される．この各マクロタスク内部にはサブマクロタスクが含まれており，そのサブマクロタスクが 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に分割しておく[11]．

本評価では，まず，Turb3d プログラムを第 6 章で用いる Sun Ultra80 上で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，62 個のマクロタスク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 情報を用いて，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，**図 3** の結果が得られる．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含まれていない．図 3 において，提案する階層統合型実行制御方式を用いた場合には，いずれのプロセッサ台数においても，プロセッサのクラスタリングは不要である．一方，従来の階層型実行制御方式を用いた場合には，プロセッサのクラスタリングが必要であり，例えば，階層型 4PC の実行の場合には，全 PE を 4 つのプロセッサクラスタに分けたクラスタリングを意味している．即ち，全 PE が 8 のときの階層型 4PC 実行では各 PC は 2PE 構成となり，全 PE 数が 4 の場合には各 PC は 1PE 構成となる．

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方式は，従来の階層型実行制御方式（可能な全てのクラスタリング）と比べて，顕著に処理時間が短縮されている．例えば，8PE 実行の場合には，階層統合型実行制御方式により，処理時間が従来の階層型 8PC 実行より約 30% 短縮されている．それゆえ，階層統合型実行制御方式は，限られたプロセッサ台数においても，全階層の並列性を効果的に利用することが可能であることが確認された．

6. 階層統合型実行制御方式の OpenMP 実装による Turb3d ベンチマークの性能評価

本章では，階層統合型実行制御方式を伴う粗粒度タスク並列処理を，SPECfp95 の Turb3d ベンチマークプログラムに対して適用し，SMP（Sun Ultra80）上で性能評価した結果について述べる．

性能評価に用いる Sun Ultra80 は，Ultra SPARC II（450MHz）プロセッサ 4 台を，Gigaplane インターコネクトにより，1GB 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．

表 2 Sun Ultra80 上での Turb3d プログラムの実行結果

実行方式	PE 数	実行時間 [s]	速度向上率
シーケンシャル実行	1	21.950	1.00
階層統合型実行制御	2	12.148	1.81
階層統合型実行制御	3	9.008	2.44
階層統合型実行制御	4	7.558	2.90

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実行している．実行結果は，**表 2** に示す通りである．

次に，提案する階層統合型実行制御を伴う粗粒度タスク並列処理の場合には，並列プログラムを OpenMP API を用いて実装した．この際，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を採用した．また，階層統合型実行制御の並列プログラムに対するコンパイルオプションは，`-fast -openmp -stackvar` となっている．この場合，4PE 上での並列処理時間は，表 2 に示す通り，シーケンシャル実行の場合の 1/2.90 に短縮されている．

また，**図 4** は，Turb3d プログラムのメインループ（62 個マクロタスクから構成される）を 12 回転分，階層統合型実行制御方式により 4PE 上で実行したトレー

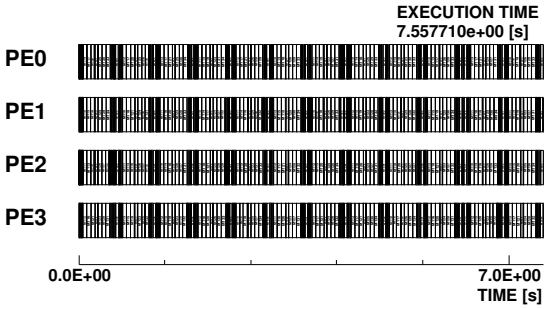

図 4 Ultra80 上での階層統合型実行制御による Turb3d（全体）の実行結果

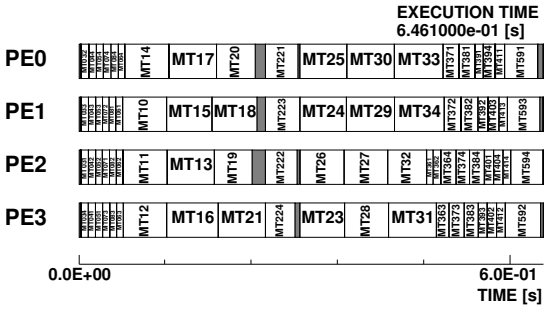

図 5 Ultra80 上での階層統合型実行制御による Turb3d（1 回転目メインループ）の実行結果

スであり，このトレースから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，プロセッサ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，処理時間の小さいマクロタスクの集合でありオーバヘッドはほとんど含まれていない．

一方，**図 5** の実行トレースは，Turb3d プログラムのメインループ（62 個マクロタスクから構成される）の第 1 回転目の実行結果（即ち図 4 の最初の部分の拡大図）を示したものである．図 5 の実行トレースから分かる通り，提案する階層統合実行制御方式は 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極めて小さいことが分かる．

7. おわりに

本稿では，粗粒度タスク並列処理における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さ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，異なる階層の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方式は，SMP 上で OpenMP のようなマルチスレッドにより実装可能であり，Turb3d プログラムの場合，異なる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行可能であることが確認された．また，シミュレーションによる性能評価では，従来の階層型実行制御方式と比べて，8PE 時に実行時間が約 30%短縮され，提案手法の有効性が確かめれた．

今後の課題としては，階層統合型実行制御方式とデータローカリティ最適化技術[11] との統合により，キャッシュ有効利用も考えられる．本研究の一部は，経済産業省/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) M. Wolfe. High performance compilers for parallel computing. *Addison-Wesley Publishing Company*, 1996.
2) R. Eigenmann, J. Hoeflinger, and D. Padua. On the automatic parallelization of the Perfect benchmarks. *IEEE Trans. on Parallel and Distributed System*, Vol. 9, No. 1, Jan. 1998.
3) M.W. Hall, J.M. Anderson, S.P. Amarasinghe, B.R. Murphy, S.-W. Liao, E. Bungnion, and M.S. Lam. Maximizing multiprocessor performance with the SUIF compiler. *IEEE Computer*, 1996.
4) A.W. Lim and M.S. Lam. Cache optimizations with affine partitioning. *Proceedings of the Tenth SIAM Conference on Parallel Processing for Scientific Computing*, 2001.
5) 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.
6) 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.
7) X. Martorell, E. Ayguade, N. Navarro, J. Corbalan, M. Gonzalez, and J. Labarta. Thread fork/join techniques for multi-level parallelism exploitation in NUMA multi-processors. *Proceedings of International Conference on Supercomputing*, 1999.
8) 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, Apr. 2002.
9) 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.
10) 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.
11) H. Kasahara and A. Yoshida. A data-localization compilation scheme using partial static task assignment. *Journal of Parallel Computing*, Vol. 24, No. 3, May. 1998.