2006－ARC－166（2）
2006／1／24

SMP クラスタ上でのタスク粒度を考慮した階層型粗粒度並列処理

角田昌芳[†]　田邊浩志[†]　本多弘樹[†]

本研究では，SMP クラスタ上で新たなタスク粒度調整手法を用いた階層型粗粒度並列処理手法を提案する．本手法では，SMP ノード内の CPU に割り当てる「下の階層」の粗粒度タスク (マクロタスク) を生成し，その後 SMP ノードに割り当てる「上の階層」のマクロタスクを生成する．本手法により，ノードに割り当てるマクロタスクの粒度を SMP クラスタの環境に合わせて調整できる．本稿では，マクロタスク内部の並列性を利用して，SMP ノードのもつ CPU 資源を有効利用する粒度調整手法を提案し，実装方法，予備評価について述べる．

Hierarchical Macro-data-flow Using Task Granularity Adjustment on an SMP-cluster

Masayoshi Tsunoda,[†] Hiroshi Tanabe[†] and Hiroki Honda[†]

To realize the hierarchical macro-data-flow processing on an SMP-cluster, we propose the method of adjusting a task granularity. This method firstly creates coarse grain tasks(macro-tasks) which are assigned to CPUs, and then creates upper layer macro-tasks, which are assigned to SMP nodes. This method adjusts a macro-task granularity in consideration of the environment of an SMP-cluster. Using a parallelism of inside of a macro-task, this paper proposes the method to adjust a macro-task granularity for effectively using CPUs of an SMP node.

1. はじめに

SMP(Symmetrical Multi-Proccessor) マシンなどの共有メモリ型並列計算機上の並列処理方式として，従来よりループ並列化が用いられており，様々な並列化技術が開発され性能向上がなされてきた．さらなる性能向上のための並列処理方式として，ループレベルの並列化に加えて，サブルーチンや代入文などの基本ブロックといった粗粒度タスク (マクロタスク) レベルの並列性を利用する粗粒度並列処理が有効であることが報告されている[1)~4)]．

粗粒度並列処理では，コンパイル時にプログラムをマクロタスク単位に分割し，マクロタスク間の並列性を実行開始条件として，マクロタスク間のデータ依存関係をデータ到達条件として求める．実行時には，実行開始条件を検査し，この条件が成立したマクロタスクから順次 CPU へ割り当て，データ到達条件[2)] を検査し必要なデータを転送することで処理を進める．

一方で SMP マシンを高速なネットワークで接続した SMP クラスタには，SMP ノード間と SMP マシン内部の CPU 間の 2 つのハードウェア階層において並列性が存在する．SMP クラスタの性能を最大限引き出すためには，多くの SMP ノードと CPU を利用して並列処理をおこなうことが重要である．

SMP クラスタ上で粗粒度並列処理をおこなう場合，プログラムのもつ階層的な並列性と SMP クラスタのもつ階層的な並列性を適切にマッピングして実行することが重要になる．粗粒度並列処理するプログラムには，マクロタスク間とマクロタスク内部の 2 階層の並列性が存在する．これら階層的な並列性を利用する一つの手法として，SMP ノード間でマクロタスク間の並列性を利用した粗粒度並列処理を用い，SMP ノード内部でマクロタスク内部のループ並列性を利用した OpenMP によるループ並列処理を用いる手法が提案されている[4)]．

また粗粒度並列処理には，マクロタスク内部をさらに細かなマクロタスクに分割し粗粒度並列処理をおこなう階層型粗粒度並列処理が提案されている[5)]．この階層型粗粒度並列処理を利用することで，ループ以外のマクロタスクも SMP ノード内部で並列化することが可能になるため，SMP クラスタ上にプログラムを適切にマッピングすることが可能であると考えられる．

従来の階層型粗粒度並列処理では，ループ，サブルーチンの内部をさらに粗粒度並列処理をする．この手法で分割されるマクロタスクの粒度は，SMP クラスタのもつ階層的な並列性を考慮せず SMP ノード数とプログラムによって決まる．このため，SMP クラスタの

† 電気通信大学 大学院情報システム学研究科
Graduate School of Information Systems,The University of Electro-Communications

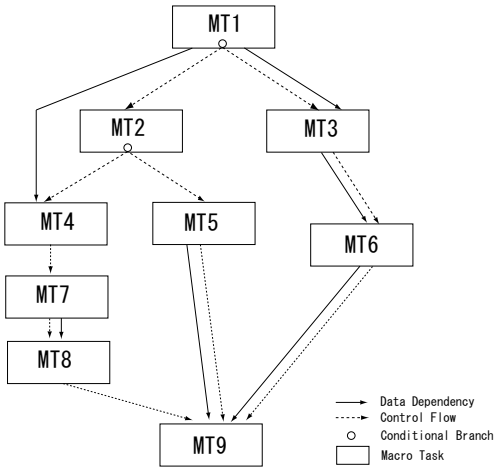

図 1 マクロフローグラフの例

もつ階層的な並列性を利用できない場合があり，SMP クラスタの性能を発揮させることが困難であった．

そこで本研究では，初めに SMP クラスタで計算に使用できる CPU 数を考慮して CPU に割り当てる「下の階層」のマクロタスクを生成する．そして，これら生成されたマクロタスクを利用して SMP ノードに割り当てる「上の階層」のマクロタスクを生成する．これにより「上の階層」のマクロタスクを生成する際にタスク粒度を調整することができるため，SMP クラスタの性能を発揮させることができる．

本稿では，SMP クラスタの多くのノード，CPU を利用して並列処理をおこなうようにタスク粒度調整手法を用いた階層型粗粒度並列処理を提案する．

2. 粗粒度並列処理[1)~4)]

2.1 マクロタスク

粗粒度並列処理では，プログラムのループ，基本ブロック，サブルーチンなど粒度の大きい処理をマクロタスクとし，マクロタスクをプロセッサへの割り当て単位として並列に処理する．

コンパイル時には，プログラムをマクロタスクに分割し，マクロタスク間の制御フローとデータ依存を**図 1** のようなマクロフローグラフで表現する．マクロタスク分割の際は，制御フローグラフが非循環になるようにする．

2.2 実行開始条件

実行開始条件とは，あるマクロタスクの実行開始が可能となる条件で，他のマクロタスクの実行状況を項とした論理式で表現したものである．この論理式は＜マクロタスク実行終了＞と＜分岐方向決定＞の 2 種類の原子条件および論理演算子 ∨(論理和) と ∧(論理積) で構成される．

2.3 データ到達条件

分散メモリ型並列計算機上での粗粒度並列処理では，データ依存のあるマクロタスクが異なるメモリ空間上のプロセスで実行された場合に，明示的なデータ転送が必要になる．このため，どのマクロタスク間でどのデータに関するデータ授受が必要かを明確にしなくてはならない．

しかしながら，あるマクロタスク MTa で MTa 以外で定義された変数 v が使用され，変数 v を定義するマクロタスクが複数ある場合，どの v の値を参照すべきかは，実行時でなければ確定できない．よって，どのマクロタスク間でどの変数に関する通信をおこなうかは実行時に判断する．

データ到達条件とは，あるマクロタスク MTa に到達する[6)] 変数 v の定義をおこなうマクロタスクの集合を S_v^a としたとき，MTa での v の値が，$MTb \subset S_v^a$ で定義した値となることが確定するための条件で，実行開始条件と同様に他のマクロタスクの実行状況を項とした論理式で表現する．

2.4 マクロタスクスケジューリング

粗粒度並列処理では，プログラムの実行時にスケジューラがマクロタスクを順次プロセッサに割り当てることで処理を進める．このスケジューラの実装には，集中型ダイナミックスケジューリング方式 (以下，集中型方式) と，分散型ダイナミックスケジューリグ方式 (以下，分散型方式) を用いることができる[7)]．集中型方式では，特定のプロセスにスケジューリングコードを実行させる．一方，分散型方式では，各プロセスにスケジューリングコードを分散させて実行させる．

ここでは，集中型方式を例として分散メモリ型並列計算機上での粗粒度並列処理におけるスケジューラの動作[2)] を説明する．スケジューラは以下の動作を繰り返し実行し，マクロタスクのスケジューリングをおこなう．

1. **実行開始条件の検査**　マクロタスクの割り当てられたプロセスから通知される＜マクロタスクの終了＞および＜分岐方向決定＞の情報を元に，実行開始条件を検査し，条件が成立したマクロタスクをレディマクロタスクとする．
2. **マクロタスクの割り当て**　所定の割り当て戦略に従いレディマクロタスク中のどのマクロタスク MTa をどのプロセス P_a に割り当てるかを決定する．
3. **データ到達条件の検査**　プロセスに割り当てることが決定したマクロタスク MTa が使用する変数 v に関するデータ到達条件を検査し，条件が成立したマクロタスク MTb と MTb が割り当てられたプロセス P_b を求める．
4. **データ転送の指示**　P_a と P_b が同一の場合，P_a で既に実行されたマクロタスクが v を参照するために P_b からのデータを受信した場合など MTb が定義した v の値が P_a に存在する場合には，デー

タ通信の指示はおこなわない．一方で，v の値が P_a に存在しない場合には，P_b に対して MT_b で定義された v の値を P_a へ送信するよう指示し，P_a には v を P_b から v 受信するように指示する．

3. SMP クラスタ上での階層型粗粒度並列処理

3.1 SMP クラスタの階層性とプログラムの階層性

SMP クラスタには，SMP ノード間と SMP マシン内部の CPU 間の 2 つのハードウェア階層において並列性が存在する．

一方で粗粒度並列処理においてプログラムをマクロタスクに分割する際，ループ，サブルーチンといったマクロタスクは，内部に並列性をもつ可能性がある．これらのマクロタスクが内部にもつ並列性は，ループ並列性やマクロタスク内部のループ間，サブルーチン間などである．

SMP クラスタとプログラムのそれぞれがもつ階層的な並列性を考慮した粗粒度並列処理として，SMP ノード間に粗粒度並列処理を利用し，SMP ノード内に OpenMP を用いたループ並列処理を利用する手法が有効であると報告されている[4]．しかし，ループ以外からマクロタスクが構成される場合にはループ並列性を利用できないため，性能向上が期待できない．

そこで本稿では，ループ内部の並列性とループ以外から構成されるマクロタスク内部の並列性を有効に利用するために，SMP ノード間の並列処理および SMP ノード内の並列処理に粗粒度並列処理を利用する 2 階層の階層型粗粒度並列処理方式を提案する．

3.2 階層性を考慮したマクロタスク

従来のマクロタスク分割手法[5] において，内部に階層的な並列性をもつマクロタスクは，ループやサブルーチンから生成される．これらマクロタスクの粒度は，SMP クラスタのもつ階層的な並列性によらず，プログラムによって一意に決まっていた．このため SMP クラスタのもつ並列性を利用することができない場合があり，SMP クラスタの性能を発揮させることが困難であった．したがって，SMP クラスタのもつ階層的な並列性を利用するためには，SMP ノード内の CPU 数と SMP ノード数を考慮して階層的な並列性をもつマクロタスクを生成する必要がある．

本研究で生成する階層的な並列性をもつマクロタスクは，SMP クラスタの階層的な並列性にあわせて 2 階層で構成する．ここで，SMP ノードに対して割り当てられるマクロタスクを第 1 階層マクロタスクと定義し，第 1 階層マクロタスク内部に存在し CPU に対して割り当てられるマクロタスクを第 2 階層マクロタスクと定義する．

従来のマクロタスク分割手法と異なり，プログラムは，初めに計算に使用する CPU 数を考慮して第 2 階層マクロタスクに分割する．その後，得られた第 2 階層マクロタスクを統合することで第 1 階層マクロタスクを生成する．この流れを**図 2** に示す．このように CPU に割り当てられる第 2 階層マクロタスクを先に生成することで，ノードに割り当てる第 1 階層マクロタスクの粒度調整を容易にすることが可能となる．

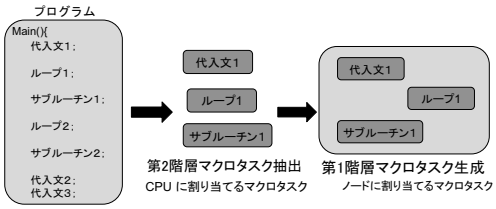

図 2 階層性を考慮したマクロタスク

3.3 階層性を考慮したマクロタスクの粒度調整手法

SMP クラスタの性能を発揮させるためには，SMP クラスタのもつ CPU を同時に多く使用すること，またノード間に発生するデータ転送を削減することが重要である．このためには，SMP ノードに割当てる第 1 階層マクロタスクの粒度を調整する必要がある．本稿では，SMP クラスタのもつ CPU を同時に多く使用するように第 1 階層マクロタスクの粒度を調整する手法を提案する．

第 1 階層マクロタスクの粒度調整は，第 2 階層マクロタスクの粒度を算出して，SMP ノード内の負荷が各ノードで均一になるように第 2 階層マクロタスクを統合することでおこなう．if 文などの分岐を含む第 2 階層マクロタスクは，他の第 2 階層マクロタスクと統合せず，一つで第 1 階層マクロタスクを形成する．第 1 階層マクロタスクの粒度調整手順は以下のようになる．

(1) 計算に使用する CPU 数を考慮して CPU 負荷が均等になるようにマクロタスク分割し，第 2 階層マクロタスクを生成する．

(2) ループイタレーション数などの静的に解析可能なパラメータを元に第 2 階層マクロタスクの粒度を算出する．

(3) 各 SMP ノードに割り当てられる計算量が等しくなるように第 2 階層マクロタスクを統合し第 1 階層マクロタスクを生成する．
このとき，if 文などの分岐を含む場合は第 2 階層マクロタスクをそのまま第 1 階層マクロタスクとする．

3.4 スケジューラの階層化

本研究では，実行開始条件とデータ到達条件を用いてスケジューリングする第 1 階層スケジューラと，実行開始条件のみを用いてスケジューリングする第 2 階層スケジューラの 2 階層のスケジューラを用いる．

第 1 階層スケジューラは第 1 階層マクロタスクをスケジューリングし，第 2 階層スケジューラは第 2 階層

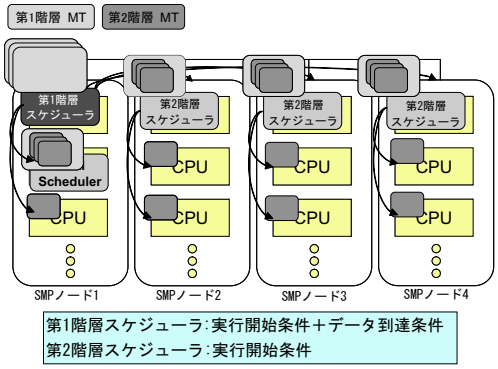

図 3　スケジューラの階層化

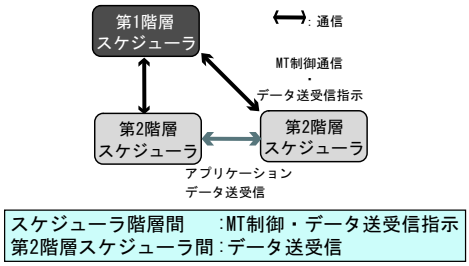

図 4　スケジューラ間の通信

マクロタスクをスケジューリングする．**図 3** に階層化されたスケジューラの構成を示す．

各階層スケジューラ間では，第 1 階層スケジューラが第 2 階層スケジューラに対しておこなう第 1 階層マクロタスクの実行指示・終了通知，第 1 階層マクロタスク間のデータ送受信の指示通信が発生する．第 2 階層スケジューラ間では，第 1 階層マクロタスク間のデータ送受信が発生する．**図 4** にスケジューラ間通信の関係を示す．

4. 階層型粗粒度並列処理の実装

ここでは，SMP クラスタ上での階層型粗粒度並列処理の実装方法について述べる．

階層型粗粒度並列処理の実装は，第 1 階層スケジューラが実行されるグローバルスケジューラプロセス (GSP) と，第 2 階層スケジューラ，アプリケーションが実行されるワーカプロセス (WP) の 2 つのプロセスから構成される．

各階層のスケジューラでのスケジューリングアルゴリズムは，CP/DT/MISF[8] を用いた．第 1 階層スケジューラには集中型方式を用いる．これは第 1 階層スケジューラを分散型方式で実行した場合，SMP ノード間でスケジューリングを同期するための通信が必要となり，スケジューリングコストが大きくなるためである．

GSP は，第 1 階層スケジューラのみが動作し SMP クラスタ中の全 CPU のうち 1 つを占有する．GSP では，第 1 階層マクロタスクのスケジューリング，第 1 階層マクロタスク実行指示通信，データ転送指示通信をおこなう．GSP は以下の動作を最後の第 1 階層マクロタスクを実行するまで繰り返す．下記動作が終了した際は，全ての WP に実行終了を通知する．

(1)　第 1 階層スケジューラを起動し第 1 階層マクロタスク $P1$ の割り当て WP を決定する
(2)　$P1$ の実行を割り当てた WP に $P1$ の実行を指示する
(3)　$P1$ のデータ到達条件を検査し必要なアプリケーションデータ転送を WP に指示する
(4)　$P1$ の終了通知を受信する

WP は，内部に第 2 階層スケジューラスレッド (WST) とアプリケーション実行スレッド (WAT) が存在し，各 SMP ノード上で動作する．WST では，データ転送と第 2 階層マクロタスクスケジューリングを逐次的におこなう．WAT では，WST から割り当てられた第 2 階層マクロタスクを実行する．WST と WAT は並列に実行される．WST は，以下の動作を最後の第 1 階層マクロタスクを実行するまで繰り返す．

(1)　第 1 階層マクロタスク $P1$ の実行を GSP から受信する
(2)　アプリケーションデータ転送指示を GSP から受信する
(3)　アプリケーションデータを指示された WP から受信または指示された WP へ送信する
(4)　必要なアプリケーションデータが全てそろった時点から，第 2 階層スケジューラが最後の第 2 階層マクロタスクを実行終了通知を受信するまで (4a)-(4b) を繰り返す
　(a)　$P1$ の第 2 階層マクロタスク $P1_s$ の実行開始条件を検査し WAT へ割り当てる
　(b)　WAT から $P1_s$ の実行終了通知を受信する
(5)　$P1$ の実行終了を GSP へ通知する

以上の処理の流れおよび通信の関係を**図 5** に示す．

5. 予備評価

提案手法を SMP クラスタ上に実装し，予備評価をおこなった．評価方法は，第 1 階層マクロタスクの生成にタスク粒度調整をした場合とタスク粒度調整をしない場合の階層型粗粒度並列処理の実行時間比較である．

評価環境は，Ultra SPARC II 450MHz を 4 つ搭載した SMP ノードを 3 台用いて構築した SMP クラスタである．**表 1** に評価に用いた SMP ノードの性能を示す．評価に用いたベンチマークプログラムは SPEC CFP95 の swim である．swim は，メインループから呼ばれるサブルーチン CALC1，CALC2，CALC3 を

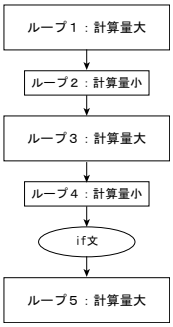

図 6　swim の構成

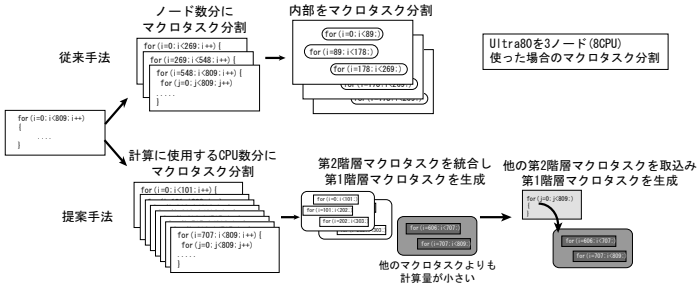

図 7　swim におけるマクロタスク生成過程の違い

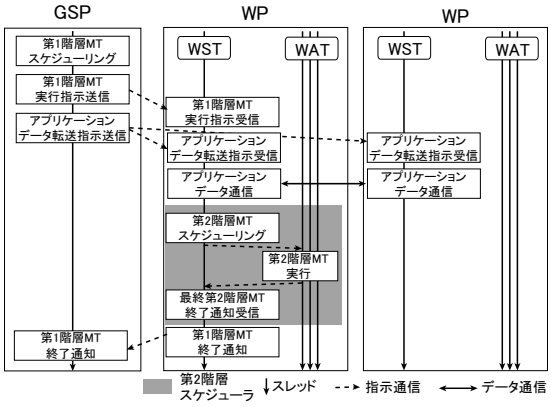

図 5　処理の流れ

表 1　評価環境

CPU	Ultra SPARC II 450MHz x 4
Memory	2GB
NIC	Ehternet 100BASE-TX
OS	Solaris9
MPI	MPICH-1.2.6
Compiler	GNU C Compiler 2.9.5

インライン展開したものを使用した．粗粒度並列処理は，メインループのイタレーション内を対象としておこなった．評価プログラムのマクロタスク分割，並列化プログラムの生成は人手でおこなった．

5.1　swim におけるマクロタスク生成

粗粒度並列処理をおこなう swim のプログラムは，**図 6** のように計算量の大きいループ 3 つと計算量の小さいループ 2 つからなる．swim の従来のマクロタスク分割手法と粒度調整をおこなうマクロタスク生成手法の違いを使用ノード数 3 の場合で説明し，**図 7** にマクロタスク分割の流れを示す．

粒度調整をしない場合には，初めに各ループを使用ノード数 3 でループ分割して第 1 階層マクロタスクを生成する．次に，得られた第 1 階層マクロタスクの内部を SMP 上で計算に使用できる最大 CPU 数 3 で分割して第 2 階層マクロタスクを生成する．

粒度調整する場合には，初めに SMP クラスタ全体で計算に使用できる CPU 数 8 でプログラム全体を第 2 階層マクロタスクに分割する．ここでは，ループ 1 における第 1 階層マクロタスク生成手法を説明する．

ループ 1 を CPU 数 8 を用いてマクロタスク分割し，第 2 階層マクロタスクを生成する．次に，生成した第 2 階層マクロタスクの計算量を算出して，各ノードで実行する第 2 階層マクロタスクの計算量の合計が最も均一になるように統合して，第 1 階層マクロタスクを生成する．ループ 1 の場合，第 1 階層マクロタスクは使用ノード数に合わせて 3 つ生成するが，それぞれの内部の第 2 階層マクロタスク数は 3:3:2 となり，第 1 階層マクロタスクの計算量は均一にならない．そこで，この計算量の小さい第 1 階層マクロタスクに，並列実行できるループ 2 の一部の第 2 階層マクロタスクを統合することで粒度調整をおこなう．

このようにして第 1 階層マクロタスク内部の並列性を増加させることで，SMP ノード内の CPU に同時に多くの第 2 階層マクロタスクを割り当てることができる．

粒度調整なしの場合は各ノードにおいてループ 1 の 269 回の最外側ループを実行する．一方で，粒度調整ありの場合は 303 回の最外側ループを実行するノードと 202 回の最外側ループとループ 2 の一部を実行するノードが存在する．このように，粒度調整なしの場合よりも粒度調整ありの場合のほうがタスク粒度が大きくなっている．

5.2　swim の実行結果

swim のデータサイズは 809x809 でメインループイタレーション回数を 100 回とした．

表 2 に swim の実行結果を示す．階層型粗粒度並列処理を用いた結果，逐次に比べて 3 ノードで 3.53 倍の速度向上が得られた．しかし，粒度調整なしの場合に比べて提案する粒度調整ありの場合の実行時間の短縮はほとんど実現することができなかった．各ループを粒度調整した結果が，粒度調整しない場合に比べて大きな違いがなかったことが原因であると考えられる．

表 2 swim の実行結果 [秒]

ノード数 (計算 CPU 数)	粒度調整なし	粒度調整あり	実行時間比
逐次 (1CPU)	302.8	-	1.00
1 ノード (2CPU)	157.2	156.3	0.99
2 ノード (5CPU)	102.4	101.4	0.99
3 ノード (8CPU)	86.4	85.7	0.99

表 3 階層型粗粒度並列処理のオーバヘッド [秒]

ノード数 (計算 CPU 数)	総実行時間	a	b	a + b
逐次 (1CPU)	302.8	-	-	-
1 ノード (2CPU)	157.2	0．08	0.15	0.23
2 ノード (5CPU)	102.4	0．25	0.53	0.78
3 ノード (8CPU)	86.4	0.47	0.98	1.45

5.3 実行時オーバヘッドの測定

SMP クラスタ上で階層型スケジューラの用いた階層型粗粒度並列処理のスケジューリングオーバヘッドが，どの程度実行結果に影響しているかを調査する．swim のプログラムを実行して，以下の 2 つのオーバヘッドを計測して**表 3** にまとめた．

(a) 第 1 階層スケジューラによるスケジューリングオーバヘッド (表 3 の a)

(b) 第 2 階層スケジューラによるスケジューリングオーバヘッド (表 3 の b)

この結果より，第 1 階層スケジューラおよび第 2 階層スケジューラによるスケジューリングオーバヘッドは，ともに 3 ノードで実行した場合に最大であった．このとき，実行時間全体に占める第 1 階層スケジューラのオーバヘッドの割合は 0.5%，第 2 階層スケジューラのオーバヘッドの割合は 1.1%であった．スケジューリングオーバヘッドを合計しても，実行時間全体に占める割合は 1.5% である．このため，スケジューリングオーバヘッドは無視できる程度であると考えられる．

6. おわりに

本稿では，SMP クラスタ上で階層型粗粒度並列処理を実現する手法を提案した．提案手法は，ループ，サブルーチンだけでなく，2 階層の並列性をもつマクロタスクを新たに生成して階層型粗粒度並列処理をおこなう処理手法である．提案手法では，2 階層の並列性をもつマクロタスクを生成する際に，CPU に割り当てる「下の階層」のマクロタスクから生成するため，SMP ノードに割り当てる「上の階層」のマクロタスクに対して様々な粒度調整が適用できるという利点がある．本稿で示したマクロタスクの粒度調整手法は，SMP クラスタのもつ CPU 資源を最大限に利用して処理をおこなう手法である．

予備評価の結果より，提案した階層型粗粒度並列処理を SMP クラスタ上に適用することが有効であることが確認できた．しかしながら，swim の実行においては，提案した粒度調整手法は粒度調整をしない場合と比較して，大きな性能向上を示すまでには至らなかった．

他のタスク粒度調整手法として，ノード間のデータ転送を最適化するようにマクロタスクを生成するデータローカライゼーション手法[9] を，SMP クラスタ上での階層型粗粒度並列処理向けに開発することが考えられる．また，SMP ノード内部でも粗粒度並列処理を用いていることから，CPU のもつキャッシュを最大限利用する最適化なども可能である[7]．

今後の課題として，データ転送の最適化を考慮した粒度調整手法の開発があげられる．提案手法では，性能向上を阻害する要因を特定するため，データ転送オーバヘッドの測定などの詳細なデータ取得や，他のアプリケーションを用いた評価をおこなう予定である．

謝辞 本研究を進めるにあたって，貴重なご意見を頂いた東京工業大学の合田憲人助教授，東邦大学の吉田明正助教授の両氏に深く感謝いたします．

本研究の一部は科学研究費 (基盤研究 (C) 16500025) によるものである．

参 考 文 献

1) APC2003: アドバンスト並列化コンパイラ国際シンポジウム資料，(財) 日本情報処理開発協会 (JIPDEC) , 早稲田大学理工学部 (2003).
2) 本多弘樹，上田哲平，深川　保，弓場敏嗣：分散メモリシステム上でのマクロデータフロー処理のためのデータ到達条件，情報処理学会論文誌: ハイパフォーマンスコンピューティングシステム，Vol.43, No.SIG6(HPS 5), pp.45–55 (2002).
3) 田邊浩志，本多弘樹，弓場敏嗣：ソフトウェア分散共有メモリを用いたマクロデータフロー処理，Vol.46, No.SIG4(ACS 9), pp.56–67 (2005).
4) 那須弘志，田邊浩志，本多弘樹，弓場敏嗣：SMP クラスタ上での MPI と OpenMP を用いたマクロデータフロー処理，情報処理学会研究報告，2004-HPC-99, pp.67–72 (2004).
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，情報処理学会論文誌，Vol.35, No.4, pp.513–521 (1994).
6) Aho, A. V., Sethi, R. and Ullman, J. D.: *Compilers - Principles, Technique, and Tools*, Addison-Wesley (1988).
7) 石坂一久，中野啓史，八木哲志，小幡元樹，笠原博徳：共有メモリマルチプロセッサ上でのキャッシュ最適化を考慮した粗粒度タスク並列処理，情報処理学会論文誌，Vol. 43, No. 4, pp. 958–970 (2002).
8) 笠原博徳：並列処理技術，コロナ社 (1991).
9) 吉田明正，前田誠司，　尾形航，笠原博徳：Fortran マクロデータフロー処理におけるデータローカライゼーション，情報処理学会論文誌，Vol.35, No.9, pp.1848–1860 (1994).