マルチグレイン並列性向上のための選択的インライン展開手法

白子 準[†] 長澤耕平[††] 石坂一久[†]
小幡元樹[†††] 笠原博徳[†]

マルチプロセッサシステムの利用範囲拡大と共に実質実行性能 (実効性能)・ソフトウェア生産性・価格性能比の向上のため，自動並列化コンパイラの必要性が高まっている．特にコンパイラによる実効性能を高めるため，基本ブロック，ループ、サブルーチン間の粗粒度並列処理・ループイタレーション間の中粒度並列処理・基本ブロック内ステートメント間の近細粒度並列処理を階層的に組合せ，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が重要となっている．マルチグレイン並列処理において階層的に並列性を抽出し，効率よい並列実行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，適切なプロセッサ数を配分する必要がある．またこの際，階層の異なる，すなわちネストされたサブルーチンをインライン展開により同一階層になるようリストラクチャリングすることで，粗粒度並列性を高めることが可能となるため，これも考慮し適切なプロセッサ配分を行なう必要がある．本稿ではプログラム中の各階層の並列度を用いマルチグレイン並列性を高めるためにインライン展開すべきサブルーチンを選択する手法とそれを考慮したプロセッサ配分法を提案する．本手法の性能を IBM RS6000 PowerPC 604e High Node 8プロセッサミッドレンジ SMP サーバ上，および 1.1GHz の Power4 を搭載した 16 way ハイエンド SMP サーバ IBM pSeries690 regattaH 上にて，SPEC95FP ベンチマークのうち，並列性の高いサブルーチンが異なる階層 (ネストレベル) に分散しているプログラムである 103.su2cor，107.mgrid， 125.turb3d を用いて評価を行った．逐次処理に対して RS6000 上で 2.84〜6.04 倍，regattaH 上で 3.54〜11.19 倍，またインライン展開を併用しない従来のプロセッサ配分手法に対して RS6000 上で 1.12〜1.79 倍，regattaH 上で 1.03〜1.47 倍の高速化が 可能になることが確かめられた

Selective Inline Expansion for Improvement of Multi Grain Parallelism

Jun shirako,[†] Kouhei Nagasawa,[†] Kazuhisa Ishizaka,[†]
Motoki Obata[†††] and Hironori Kasahara [†]

With the increase of applications of multiprocessor systems, needs of automatic parallelizing compilers are increasing to improve effective performance, cost performance, and software productivity. Especially, for higher effective performance by compiler, a multi-grain parallel processing which exploits coarse grain parallelism among loops, subroutines and basic blocks, medium grain parallelism among loop-iterations and near fine grain parallelism among statements inside a basic block, is getting important. In multi-grain parallel processing, it is required to assign the appropriate number of processors to each nested layer, considering the parallelism of each layer. At that time, inline expansion of subroutines having large parallelism in a lower layer can increase coarse grain parallelism significantly. Therefore, considering this program restructuring, a compiler must assign processors to each layer. To this end, this paper proposes a selective inline expansion scheme for improvement of multi grain parallelism. Effectiveness of the proposed scheme is evaluated on IBM RS6000, midrange SMP server with 8 processors and IBM pSeries690 regattaH, highend SMP server with 16 processors, using 103.su2cor, 107.mgrid, 125.turb3d of SPEC95FP. The multi grain parallel processing using the proposed scheme gave us 2.84 to 6.04 times speedup on RS6000, 3.54 to 11.19 times speedup on regattaH against sequential processing, 1.12 to 1.79 times speedup on RS6000, 1.03 to 1.47 times speedup on regattaH against conventional multi-grain parallelization.

† 早稲田大学理工学部コンピュータ・ネットワーク工学科
Department of Computer Science, Waseda University
†† 富士通株式会社
Fujitsu Limited
††† 株式会社日立製作所
Hitachi, Ltd.

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においては，従来よりループ並列化手法[1] が用いられている．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] などの

ようなループ並列化コンパイラでは，レンジテスト，シンボリック解析，インタープロシージャ解析等の強力なデータ依存解析手法と，アレイプライベタゼーション，アフィンパーショニング等のループリストラクチャリング手法を組み合わせることで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は既に成熟期に入っており，今後大幅な性能向上は見込めないと考えられている．したがって，マルチプロセッサシステムの性能向上のためには，これまで用いられてきたループ並列化技術に加え，サブルーチン・ループ・基本ブロックの間の粗粒度タスク並列処理，またチップマルチプロセッサ等で有効な基本ブロック内ステートメント間近細粒度並列処理の階層的な利用が重要となっている．このようなマルチグレイン並列処理[4)~6)]ではプログラムを粗粒度タスク (マクロタスク，MT)[4),6)] に分割し，最早実行可能条件解析[6),7)] を用いて各マクロタスク間の並列性を抽出してマクロタスクグラフ (MTG)[4),6)] を生成する．生成されたマクロタスクがサブルーチンブロック，ループブロックの場合は，階層的にその内部をマクロタスクに分割し，プログラムの各ネストレベルに対してマクロタスクグラフを定義し，階層的なマクロタスクグラフを生成することによりプログラム全域の並列性を抽出する．

マルチグレイン並列処理においては，マクロタスクグラフの並列性や形状に応じ，各階層に割り当てるプロセッサ数やどの粒度で並列処理するかを適切に決めなければならない．この目的のため，並列処理階層自動決定手法[8),9)]ではインライン展開は考慮せずプログラムの各階層の並列性を有効に利用するプロセッサ数や並列処理方式を決定する．しかし，実際のプログラムでは，内部に高い並列性を含むサブルーチンが異なる階層に分散している場合が多く，インライン展開によるプログラム階層間での並列性の移動を含めた総合的な並列性の利用を考える必要がある．

サブルーチン間の並列性解析，抽出については従来よりループ内のサブルーチンコールを解析しそのループを並列化するためのインタープロシージャ解析 (IPA) 手法が研究されている．例えば，配列添字の詳細とループ範囲をコールサイトに伝搬するアトムイメージ法[10)]や，ループ境界とループストライドからサブルーチン内での参照配列部分を決定しコールサイトへ伝搬する Bounded Regular Section 解析法[11)] 等がある．また，スタンフォード大学の SUIF コンパイラでは，サブルーチンを選択的にクローニング[12)]し，コールサイトコンテキストの相違による解析精度の低下を防いでいる．本論文は推定したプログラム中の全階層の並列度を用いてマルチグレイン並列性を効果的に利用することを目的としている．各階層におけるマクロタスクグラフ上のサブルーチンに対してインライン展開を行ない上位階層の並列性を高めるべきか，またはサブルーチン内部の並列性をインライン展開を適用せずに利用すべきかを判断する選択的インライン展開手法を提案し，共有メモリ型マルチプロセッサシステム上で性能を評価した結果について述べる．

2. 粗粒度タスク並列処理

本章では，逐次プログラムを階層的に粗粒度タスク分割して階層的マクロタスクグラフを生成し，粗粒度タスク間並列性解析を行なう手法について述べる．

粗粒度タスク並列処理とは生成されたマクロタスクをプロセッサエレメント (PE)[9)]，もしくはプロセッサグループ (PG)[9)] に割り当てて実行することによりマクロタスク間の並列性を利用する並列処理手法である．

2.1 粗粒度タスク生成

粗粒度タスク並列処理では，プログラムは基本ブロックまたはその融合ブロックで構成される疑似代入文ブロック BPA[4)]，DO ループや後方分岐により生じるナチュラルループで構成される繰り返しブロック RB[4)]，サブルーチンブロック SB[4)] の 3 種類のマクロタスク MT[4),6)] すなわち粗粒度タスクに分割される．繰り返しブロック RB やサブルーチンブロック SB に対しては，その内部をさらに階層的にマクロタスク分割し階層的な並列性を利用する．

2.2 粗粒度並列性抽出

マクロタスク生成後，各階層においてマクロタスク間のデータ依存と制御依存をアトムイメージ法など従来の手続き間解析法なども用いて可能な限り解析し，階層的マクロフローグラフ[4),6)] を生成する．

次に，マクロタスク間の並列性を抽出するために各マクロタスクの最早実行可能条件[6),7)] を解析し，階層的なマクロタスクグラフ MTG[4),6)] を生成する．ここで最早実行可能条件とは，制御依存とデータ依存を考慮したマクロタスクの最も早く実行を開始して良い条件でありタスク間の粗粒度タスク並列性を表す．

2.3 プロセッサグループとプロセッサエレメント

階層的に定義されたマクロタスクグラフを効果的に処理するためにはプロセッサも階層的にグルーピングする必要がある．本手法では，複数のプロセッサエレメント PE[9)] をソフトウェア的にグループ化したプロセッサグループ PG[9)] を定義し，このプロセッサグループにマクロタスクグラフ上のマクロタスクを割り

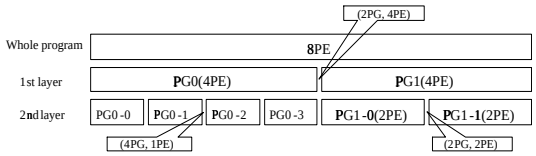


図 1 プロセッサグループ・プロセッサエレメントの階層的定義
Fig. 1 Hierarchical definition of processor groups and processor elements

当てる．割り当てられたマクロタスク内で更にマクロタスクグラフが定義されている場合は，内部マクロタスクグラフに対してプロセッサグループ内のプロセッサを階層的にグルーピングする．これを階層的に行なうことでプログラム全域にわたる階層的粗粒度タスク並列性を利用することができる．

図 1 にプロセッサグループ・プロセッサエレメントの階層的な定義の例を示す．図中，第 1 階層では (2PG, 4PE) の構成をとり 2 並列で粗粒度タスク並列処理を行ない，4 PE を割り当てられた第 2 階層ではそれぞれ (4PG,1PE)，(2PG, 2PE) という構成をとっている．このように階層的なプロセッサ構成を定義することでそれぞれの階層の並列性に適したグルーピングを行なうことが可能となる．

2.4 プロセッサグループへのマクロタスク割り当て

上述のように各階層のマクロタスクグラフに対して生成されたプロセッサグループがそのマクロタスクグラフ上のマクロタスクを処理する単位となり，次にスケジューラがマクロタスクを各プロセッサグループに割り当てる．マクロタスクグラフ上に条件分岐が無い場合はコンパイル時に静的にスケジューリングが行われ各プロセッサグループの処理するマクロタスクが決定される．マクロタスクグラフが条件分岐を含む場合は実行時にスケジューリングを行なう動的スケジューリングルーチンをコンパイラが自動生成し，並列プログラム中に埋め込まれたこのスケジューリングルーチンが実行時にマクロタスクをプロセッサグループに低オーバーヘッドで割り当てる．

以下の章においては，MT はマクロタスク，MTG はマクロタスクグラフ，PG はプロセッサグループ，PE はプロセッサエレメントを表わす．また疑似代入文ブロックを BPA，繰り返しブロックを RB，サブルーチンブロックを SB と表記する．

3. 並列処理階層自動決定手法

マルチグレイン並列処理を階層構造を持つプログラムに効果的に適用するには、各階層の並列性を解析し、その並列性に見合った PG 数・PE 数を判断することが必要である．本章では各階層の PG 数・PE 数を決定するための並列処理階層自動決定手法を述べる．本手法では，

・上位階層に存在する並列性の優先的利用
・粗粒度タスク並列性の優先的利用

の 2 点を PG 数・PE 数決定時の基本方針としている．上位階層の並列性の優先利用とは並列性がネストしている場合に，より上位階層，すなわちネストレベルの浅い階層の並列性を優先して利用することである．これは上位の階層ほどループやサブルーチンといった MT の粒度が粗く，並列処理の際の同期・スケジューリングオーバーヘッドを相対的に小さく抑えることができるためである．また粗粒度タスク並列性の優先も同様の理由であり，ループやサブルーチン間のタスク並列性といったより粒度の粗い並列性を利用することで，オーバーヘッドの小さい効果的な並列処理の実現を目指すためである．

3.1 並列度の算出

並列処理階層自動決定手法では，各 MTG の逐次処理コストとクリティカルパス長 (CP 長:入り口ノードから出口ノードまでの最長のパス長) を用いて各階層の並列度を推定する．MTG の逐次処理コストとは，内部 MT の逐次処理コストを条件分岐確率を考慮し，制御フローに従って総和したものである．今回は条件分岐確率については全分岐方向で等確率と推定している．また各 MT の処理コストの算出については，疑似代入文ブロック BPA の逐次処理コストは対象とする実機毎の各演算命令の処理コストの総和として計算し算出，繰り返しブロック RB の場合は内部 MTG の処理コストに回転数を乗じたものを逐次処理コストとしている．ここで回転数が定数でないときは RB 内で使用される配列の各次元のうち，ループインデックスを添字式中に含む次元の宣言サイズを回転数として推定する．またサブルーチンブロック SB の逐次処理コストはサブルーチン内の MTG の逐次処理コストとしている．各命令の処理コスト，条件分岐確率，回転数の推定において実行プロファイルを用いることでより推定精度を上げることが可能であるが，今回は初見のプログラムに対する性能評価を前提としたためプロファイル情報を用いず上述のような推定を行なっている．このため処理コストは必ずしも正確な値となるわけではないがこれらのコストは PG 数・PE 数の決定のために用いる値であり，プロセッサのタスクスケジューリングはスケジューリングフェーズでより詳細におこなわれるため，PG 数・PE 数決定フェーズにおいては高い厳密性は要求されない．これらの逐次処理

コストを用いて並列度を以下のように求める．MTG_i の逐次処理コストを Seq_i，MTG_i の CP 長を CP_i として，粗粒度タスク並列度 $Para_i$ を

$$Para_i = Seq_i \ / \ CP_i$$

と定義する．$Para_i$ は，MTG_i の逐次処理コスト Seq_i を最小の並列処理時間である CP_i で割ることにより MTG_i のタスク間の並列性を表わし，$\lceil Para_i \rceil$ は MTG_i を CP_i で処理するために必要な PG 数となる．

次に，粗粒度タスク並列性とループ並列性を考慮した総合的な並列性を表す $Para\_ALD$ ($Para\ After\ Loop\ Division$) を定義する．ループ並列性を粗粒度タスク並列性に変換するため，まずイタレーション間の並列処理が可能な RB に対してスケジューリングオーバーヘッド，同期オーバーヘッド等を使用する実機ごとに測定し，これらを考慮した場合の並列処理効果が期待される最小のタスク処理コスト T_{min} を求める．ここで T_{min} の値はプロセッサ数等の影響を受けるが，今回はタスク数，プロセッサ数を変化させ同期やスケジューリングのオーバーヘッドを測定し，この平均値の 4 倍を T_{min} として設定した．この T_{min} を上回るようなコストを持つ小ループに並列処理可能 RB を分割し，各小ループを粗粒度タスクと定義した際の粗粒度タスク並列性を等価粗粒度並列性と呼ぶ．また MTG_i 上に存在する並列処理可能な RB をコストが T_{min} となるよう分割したと想定した際の，MTG_i の CP 長を CP_ALD_i とし

$$Para\_ALD_i = Seq_i \ / \ CP\_ALD_i$$

と定義する．$Para\_ALD_i$ は MTG_i 上の並列処理可能 RB のイタレーション間並列性を粗粒度タスク並列性に変換してできたマクロタスクグラフに対し，$Para_i$ と同様にタスク間並列性を表わしたものである．

$Para$，$Para\_ALD$ は MTG が持つ並列性の大きさを表す指標であるが，注目している MT_i と，その内部にネストされている全ての MTG がもつ並列性の最大値を表すために，階層的最大並列度

$$Hierarchical\_Para\_max_i = Converted\_Loop\_Para_i \times \lceil Para_i \rceil \times \max_j(Hierarchical\_Para\_max_{i\_j})$$

を導入する．ここで $Converted\_Loop\_Para_i$ は，MT_i が並列処理可能 RB の場合はその等価粗粒度並列性となりイタレーションレベルの並列性を表わし，MT_i が並列処理可能 RB 以外の場合は 1 となる．

$\max_j(Hierarchical\_Para\_max_{i\_j})$ は MTG_i 上の j 番目のマクロタスク MT_{i_j} のうち最大の $Hierarchical\_Para\_max$ であり，$\lceil Para_i \rceil \times \max_j(Hierarchical\_Para\_max_{i\_j})$ は MTG_i における並列性の最大値を表わしている．これらを乗ずることで，本手法において $Hierarchical\_Para\_max_i$ は MT_i に割り当てて効果のあるプロセッサ数の上限を表わすものとしている．

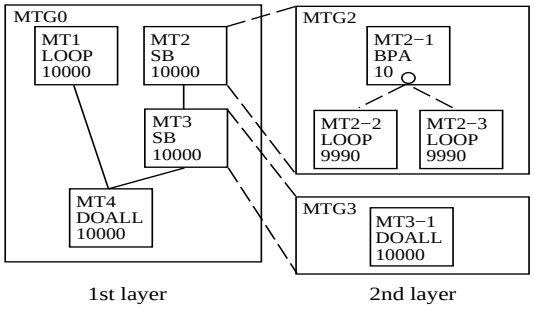


図 2 プロセッサ割り当ての例
Fig. 2 Hierarchical MTG for assignment of processors

3.2 並列度による PG 数・PE 数の決定

次に粗粒度タスク並列度 $Para$，粗粒度タスク並列性とループ並列性の総合並列度 $Para\_ALD$，階層的最大並列度 $Hierarchical\_Para\_max$ を用いて，以下のように各階層に割り当てる PG 数・PE 数 の組合わせを決定する．

step.1 並列度による初期解の決定

本手法では MTG_i の粗粒度タスク並列性を十分に生かすための PG 数 N_{PGi}，PE 数 N_{PEi} を

$$\lfloor Para_i + 0.5 \rfloor \leq N_{PGi} \leq \lfloor Para\_ALD_i + 0.5 \rfloor \quad (1)$$

かつ

$$N_{PGi} \times N_{PEi} = N_{Avail\_PEi} \quad (2)$$

を可能なかぎり満たすように決定する．

ここで N_{Avail_PEi} は MTG_i の処理に利用可能な総プロセッサ数である．(2) 式については，条件を満たす N_{PGi}，N_{PEi} のうち N_{PGi} が最大となる組合わせを選ぶ．

step.2 (1) 式を満たせない場合の N_{PGi} の決定

- $N_{Avail\_PEi} < \lfloor Para_i + 0.5 \rfloor$ の場合
  $(N_{PGi}, N_{PEi}) = (N_{Avail\_PEi}, 1)$ とする．
  ただし N_{Avail_PEi} は MTG_i の処理に利用可能な総プロセッサ数
- N_{PGi} が (1) の範囲内に存在しない場合
  $\lfloor Para_i + 0.5 \rfloor < N_{PGi}$ を満たす最小の N_{PGi} を選択する．

step.3 オーバーヘッド削減のための N_{PEi} 補正

$N_{PEi} > \max_j(Hierarchical\_Para\_max_{i\_j})$

である場合は $N_{PEi} = \max_j(Hierarchical\_Para\_max_{i\_j})$ として N_{PEi} を補正する．ただし MT_{i_j} は MTG_i 上の j 番目のマクロタスクを表し，$Hierarchical\_Para\_max_{i\_j}$ は MT_{i_j} の階層的最大並列度である．これを上位階層から順に繰り返してい

表 1 図 2 における各階層の並列度
Table 1 Parallelism of each layer in fig.2

	$Seq.cost[u.t]$	$CP[u.t.]$	$CP\_ALD[u.t.]$	$Para$	$Para\_ALD$	H_Para_max	(PG, PE)
MTG_0	40000	30000	21000	1.33	1.90	20	(2PG, 2PE)
MTG_2	10000	10000	10000	1.00	1.00	1	(1PG, 1PE)
MTG_3	10000	10000	1000	1.00	10.00	10	(2PG, 1PE)

き，全階層の PG 数，PE 数を決定する．

上記の各 step について以下に説明する．まず **step.1** では (1) 式中の $Para_i$, $Para\_ALD_i$ を整数値に変換している．(2) 式において N_{PG_i} が最大となる組合わせを選択しているが，これは一般に上位階層 MTG 上に存在する MT は下位階層の MT に比べて処理コストが大きく，一方スケジューリングや同期のオーバーヘッドは階層に関わらずほぼ一定であるため上位階層の粒度の粗いタスク間での並列処理は相対的にオーバーヘッドが低くなるためである．このため本手法では上位階層の並列性を優先する方針をとっている．

$Para_i$, $Para\_ALD_i$ の値によっては (1) 式が満たされない場合があり，**step.2** によって N_{PG_i} 数の決定を行なう．利用可能なプロセッサ数 $N_{Avail\_PE_i}$ よりもタスク間並列性 $Para_i$ の方が大きい場合はタスク間並列性を最大限に利用する．また $N_{Avail\_PE_i} = 8$ のとき N_{PG_i} のとり得る値は 1, 2, 4, 8 であるが，$Para_i = Para\_ALD_i = 6$ であるというような $Para_i$, $Para\_ALD_i$ の間に N_{PG_i} が存在しない場合は，タスク間並列性の有効利用を保証するため $\lfloor Para_i + 0.5 \rfloor \leq N_{PG_i}$ を優先する．このため上述の例では $N_{PG_i} = 8$ が選択される．

step.3 では階層的最大並列度 $Hierarchical\_Para\_max$ よりも割り当てられるプロセッサ数が大きい場合，割り当てプロセッサ数を抑制している．N_{PE_i} が MT_{i_j} に割り当てられる総プロセッサ数となるが，$Hierarchical\_Para\_max_{i\_j}$ が MT_{i_j} に割り当てて効果のあるプロセッサ数の上限を表し，このうち最大の値である $\max_j(Hierarchical\_Para\_max_{i\_j})$ が割り当てプロセッサ数の上限である．よって $Para\_max_{i\_j}$ を超えないように N_{PE_i} を補正する．

3.3 プロセッサ割り当ての例

以下では，図 2 の簡単な階層的 MTG に対してプロセッサ割り当てを行なう場合を例に取り本手法を説明する．図 2 中 の各ノードが MT，実線のエッジがデータ依存，点線のエッジが制御依存を表し，LOOP は並列処理不可能な逐次 RB，DOALL は並列処理可能な RB を表している．ここでは $T_{min} = 1000[u.t.]$ とし，第 1 階層の MTG 上の MT_1，MT_4，第 2 階層 MT である MT_{2_2}，MT_{2_3}，MT_{3_1} の内部には並列性がないものとする．この場合の各並列度を表 1 に示す．ここで [$u.t.$] は単位時間 Unit Time の略で本論文におけるタスク処理コストの単位であり，CPU の 1 クロックを 1[u.t.] としている．

ここでは MTG_0 で使用可能なプロセッサ数を 4 とした場合の各 MTG へのプロセッサ割り当て法を説明する．まず MTG_0 について考えると，**step.1** より $1 \leq N_{PG0} \leq 2$ かつ $N_{PG0} \times N_{PE0} = 4$ となり $N_{PG0} = 2$，$N_{PE0} = 2$ が選択される．次に **step.3** において $N_{PE0} = 2$ を MTG_0 の下位階層に割り当てるのが適切かどうかの判断を行なうが，表 1 における $Hierarchical\_Para\_max_3 = 10$ より MT_3 の階層的最大並列度が 10 であるため，2PE を下位階層に割り当ててもプロセッサがアイドルになる可能性は低く適切と判定される．これにより MTG_0 は (2PG, 2PE) 構成で処理するのが適切と判断される．また DOALL ループである MT_4 は MTG_0 に割り当てられる 2PG に負荷を分散するため 2 分割され，分割により生成される 2 ループをそれぞれ MT_{4a}，MT_{4b} と表わす．次に MTG_2 の PG 数･PE 数の組合わせであるが 2 プロセッサが割り当てられ，また $Para\_ALD_2 = 1$ であるため **step.1** において $N_{PG2} = 1$ が選択され，(1PG, 2PE) という組合わせが選択される．しかし MTG_2 上の MT は $Hierarchical\_Para\_max = 1$ であるので **step.3** よりこれらの MT には 1 プロセッサしか割り当ててはならないと判断され $N_{PE2} = 1$ と補正され，(1PG, 1PE) という組合わせが自動的に選ばれる．MT_1 の内部の MTG である MTG_1(図 2 中では省略) も同様に (1PG, 1PE) となり 1 プロセッサで逐次処理される．また MTG_3 については 2PE 割り当てられ **step.1** より $Para_3 = 1 < N_{PG3} < Para\_ALD_3 = 10$ となり，$N_{PG3} = 2$ が選択され (2PG, 1PE) となる．また DOALL ループである MT_{3-1} は上述の MT_4 のように 2 分割される．以上のように決定された PG 数･PE 数の組合わせを表 1 の (PG, PE) 欄に示す．

次に上記の効果を説明するため図 2 を 4 プロセッサを用いて，ループ並列処理のみを用いた場合とマルチグレイン並列処理を用いた場合の処理時間を比較する．まずループ並列処理のみの場合，MT_1 と MT_2 はループ並列処理できる箇所がないため処理時間は逐次処理時間

と同じである．MT_3 内の MT_{3-1} と MT_4 はDOALLなので4プロセッサで並列処理され，全体の処理時間は $10000+10000+10000/4+10000/4 = 25000[u.t.]$ となる．これに対しマルチグレイン並列処理を用いた場合は一方の PG(PG0 とする) が MT_2，MT_3，MT_{4a} (逐次処理コストは 5000) を処理している間に，もう一方の PG(PG1 とする) が MT_1 と MT_{4b} を処理するため，全体の処理時間は PG0 の処理時間の $10000+10000/2+5000/2 = 17500[u.t.]$ であり，ループ並列処理のみを用いた場合に比べ 7500[u.t.] 処理時間が短縮可能とコンパイラは推定する．

4. 並列性向上のためのインライン展開手法

3 章で述べたマルチグレイン並列処理ではループ並列性と粗粒度タスク並列性を階層的に利用することが可能である．しかしプログラムの並列性すなわち階層的 MTG の形状によっては上述の手法のように上位階層の並列性を積極的に利用すると下位階層の並列性を効果的に利用するのに必要なプロセッサ数を確保できなくなる場合がある．このように下位階層に並列性の高いサブルーチンブロック SB が存在するときは，この SB を上位階層にインライン展開することで上位階層の粗粒度タスク並列性を高め，プロセッサをより有効に使用することができる．以下では，より効果的にマルチグレイン並列性を抽出するための選択的インライン展開手法を提案する．

4.1 MT が内包する並列度

3 章で MT が内包する全並列性を十分に利用するプロセッサ数の上限値として階層的最大並列度 $Hierarchical\_Para\_max$ を定義した．これに対して，MT 内の全並列性を利用するために必要なプロセッサ数の下限値を表すため階層的並列度 $Hierarchical\_Para$ を以下のように定義する．MT_i 内の全下位階層に対して，並列処理可能 RB を前出の T_{min} を超えるタスクコストとなるようタスクへ分割し，各 MTG の CP 長を下位階層から順に総和したものを $Hierarchical\_CP_i$ とする．つまり MTG_i 上の j 番目のマクロタスクを MT_{i_j} として，MT_{i_j} の $Hierarchical\_CP_{i\_j}$ の和が最長となるパス長を $Hierarchical\_CP_i$ とする．ここで MT_i 自体が並列処理困難な RB の場合，その回転数を MTG_i の CP 長に乗じたものを MT_i の $Hierarchical\_CP_i$ とする．MT_i が並列処理可能 RB の場合で，MTG_i の CP 長が T_{min} を上回るコストを持つ場合は，その CP 長を $Hierarchical\_CP_i$ とし，T_{min} を上回らない場合は，T_{min} を $Hierarchical\_CP_i$ とする．

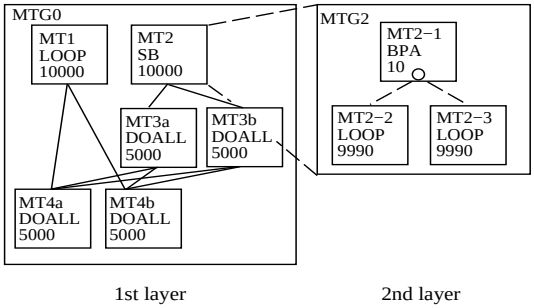


図 3 インライン展開後の階層
Fig. 3 Hierarchical MTG after inlining

$Hierarchical\_CP_i$ は MT_i 内部の階層構造を考慮した CP 長であり，これを用いて，

$Hierarchical\_Para_i = Seq_{MT_i} / Hierarchical\_CP_i$

と定義する．式中の Seq_{MT_i} は MT_i の逐次処理コストである．$Hierarchical\_Para_i$ は MT_i の逐次処理コスト Seq_{MT_i} を，階層的な並列処理を考慮した最小並列処理時間 $Hierarchical\_CP_i$ で割ることにより MT_i の階層的並列度を表わすと考え，以下の議論を行なう．

次に MTG_i に対して，インライン展開による並列性の移動を考慮した並列度 $Para\_inl\_ALD_i$ を定義する．ここで，MTG_i 上とその下位階層マクロタスクグラフ上の全 SB をインライン展開し，MTG_i 上に存在する並列処理可能 RB についてはループ分割による等価粗粒度並列性を考慮した場合の CP 長を $CP\_inl\_ALD_i$ とする．これを用いて，

$Para\_inl\_ALD_i = Seq_i \ / \ CP\_inl\_ALD_i$

と定義する．$Para\_inl\_ALD_i$ は MTG_i 上およびその下位階層に存在する全 SB をインライン展開することで，下位階層の並列性を MTG_i のネストレベルまで持ち上げた場合の粗粒度タスク並列性とループ並列性の総合値を表わす．

4.2 並列度に基づく選択的インライン展開

本手法では 4.1 節で定義した階層的並列度 $Hierarchical\_Para$，インライン展開を考慮した粗粒度タスク並列性とループ並列性の総合値 $Para\_inl\_ALD$ をもとに，3 章で述べた並列処理階層自動決定手法により割り当てられたプロセッサ数よりも多くのプロセッサを必要とする SB のみをインライン展開する．これにより不必要なインライン展開を避けることが可能であり，MTG 上の MT 数や条件分岐の増加によるスケジューリングオーバーヘッドや MTG の負荷バランスの悪化を抑えることができる．以下にインライン展開すべきサブルーチンを選択する手順を述べる．

step.1 インライン展開適用候補の選択

表 2 図 3 における各階層の並列度
Table 2 Parallelism of each layer in fig.3

	$Seq.cost[u.t.]$	$CP[u.t.]$	$CP\_ALD[u.t.]$	$Para$	$Para\_ALD$	H_Para_max	(PG, PE)
MTG_0	40000	30000	12000	1.33	3.33	20	(2PG, 2PE)
MTG_2	10000	10000	10000	1.00	1.00	1	(1PG, 1PE)

プログラムの上位階層から順に並列処理階層自動決定手法を適用し，$N_{PGi} \geq 2$ となる MTG_i を見つける．次にその MTG_i での PE 数 N_{PEi} よりも多くのプロセッサ数を必要とする SB が存在するとき，つまり

$Hierarchical\_Para_{i\_j} > N_{PEi}$

を満たす SB(ここでは MT_{i_j} は MTG_i 上の SB である MT を表わすものとする．) が存在する場合は，MT_{i_j} はインライン展開が有効であると判断し，この MTG_i をインライン展開適用階層の候補とする．

step.2 展開を考慮した PG 数・PE 数の推定

展開の候補となった階層 MTG_i 上に存在する MT_{i_j} をインライン展開した場合，MTG_i の $Para_i, Para\_ALD_i$ は変化する可能性がある．このためインライン展開後の MTG_i の並列性に応じた PG 数・PE 数を再度決定する必要があり，以下の方法より展開後の PG 数の予測値 N'_{PGi}・PE 数の予測値 N'_{PEi} を定める．

$Para_i \leq N'_{PGi} \leq Para\_inl\_ALD_i$

かつ

$N'_{PGi} \times N'_{PEi} = N_{Avail\_PEi}$

N'_{PGi}，N'_{PEi} は予測値であり，実際の PG 数・PE 数は展開後に再度並列処理階層決定手法を用いて定める．

step.3 インライン展開階層の決定

MTG_i 上の SB のうち，**step.2** で決定された N'_{PEi} よりもプロセッサを必要とし，インライン展開により MTG_i での並列性を有効に向上させる可能性のある SB を選択する．すなわち

$Hierarchical\_Para_{i\_j} > N'_{PEi}$ (1)

かつ

$Para\_inl\_ALD_{i\_j} \geq 2$ (2)

を満たす全ての MT_{i_j} をインライン展開する SB として選択する．上の式で表わされた条件のうち，(1) 式は MT_{i_j} の階層的並列度が割り当てられるプロセッサ数よりも大きいかを表わし，(2) 式はインライン展開により MTG_i での並列性を十分向上させられるかを表わしている．

step.1 から step.3 を上位階層より順に全ての階層に適用した後，インライン展開すると判定された SB に対して実際にインライン展開を行なう．これにより深い階層に存在するインライン展開されるべきサブルーチンを適切に選択することができる．インライン展開の後，再び並列処理階層自動決定手法を用いて，各 MTG の最終的な PG 数，PE 数を決定する．

4.3 インライン展開の例

以下では図 2 で表される階層的 MTG に，提案する選択的インライン展開手法を適用する例をとりながら手法を具体的に説明する．まず各階層の $Hierarchical\_Para$，$Para\_inl\_ALD_{i\_j}$ を算出する．MTG_2 では，内部 MT に並列性がないため $Hierarchical\_CP_2 = CP\_inl\_ALD_2 = Seq.cost_2 = 10000[u.t.]$ であり，$Hierarchical\_Para_2 = Para\_inl\_ALD_2 = 10000/10000 = 1$ となる．MTG_3 については内部 MT は DOALL である MT_{3-1} のみであり，等価粗粒度並列性を考慮して $Hierarchical\_CP_3 = CP\_inl\_ALD_3 = CP\_ALD_3 = 1000[u.t.]$，よって $Hierarchical\_Para_3 = Para\_inl\_ALD_3 = 10000[u.t.]/1000[u.t.] = 10$ となる．また MT4 は内部階層であるループボディにおいては並列性がないが，MT_4 自体の等価粗粒度並列性を考慮して $Hierarchical\_CP_4 = CP\_inl\_ALD_4 = CP\_ALD_4 = 1000[u.t.]$ となる．次に MTG_0 について各並列度を求める．まず $Hierarchical\_CP_0$ は MT_2，MT_3，MT_4 の $Hierarchical\_CP$ の総和であるので

$Hierarchical\_CP_0 = 10000 + 1000 + 1000 = 12000[u.t.]$，同様に $CP\_inl\_ALD_0$ も MT_2，MT_3，MT_4 の CP_inl_ALD の総和であり $CP\_inl\_ALD_0 = 10000 + 1000 + 1000 = 12000[u.t.]$ である．以上より $Hierarchical\_Para_0 = 40000[u.t.]/12000[u.t.] = 3.33$，$Para\_inl\_ALD_0 = 40000[u.t.]/12000[u.t.] = 3.33$ となる．

次にインライン展開すべき SB の選択を行なう．3 章と同様に MTG_0 では 4 プロセッサ利用な場合を考える．**step.1** に従い MTG_0 について並列処理階層決定手法を用いると，表 1 のように (2PG, 2PE) という構成が選択され MTG_0 上の MT に割り当てられる総プロセッサ数は 2 となる．一方 MT_3 が必要とするプロセッサ数は $Hierarchical\_Para_3 = 10$ であり，2 プロセッサ以上必要である．これより MTG_0

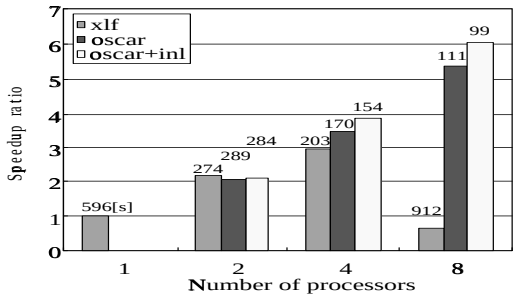


図 4 8 プロセッサまでの su2cor の速度向上率 (RS6000)
Fig. 4 speedup ratio of su2cor up to 8PE (RS6000)

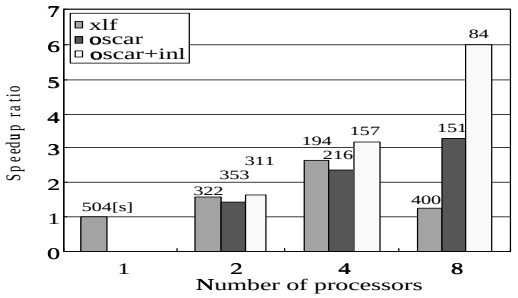


図 5 8 プロセッサまでの mgrid の速度向上率 (RS6000)
Fig. 5 speedup ratio of mgrid up to 8PE (RS6000)

はインライン展開適用階層の候補と選択される．次に **step.2** によりインライン展開後の PG 数・PE 数の予測値 $(N_{PG_i'}, N_{PE_i'})$ を決定する．$Para_0 = 1.33$ と $Para\_inl\_ALD = 3.33$ から $1 \leq N_{PG_0'} \leq 3$ かつ $N_{PG_0'} \times N_{PE_0'} = 4$ を満たす $(N_{PG_i'}, N_{PE_i'})$ の組合わせを考え，$N_{PG_0'} = 2$，$N_{PE_0'} = 2$ が選択される．よってインライン展開後に MTG_0 上の MT へ割り当てられるプロセッサ数の予測値は 2 となる．MTG_0 上の SB は MT_2 と MT_3 であるが，**step.3** において $Hierarchical\_Para_2 = 1$ より MT_2 は 2 プロセッサは必要でなくインライン展開不要と判定される．一方 MT_3 は $Hierarchical\_Para_3 = 10$ より 2 以上のプロセッサを必要とし，$Para\_inl\_ALD_3 = 10$ よりインライン展開することで MTG_0 の並列性を増加可能であり，MT_3 のインライン展開は有効と判定される．

以上の手順で指定された MT_3 をインライン展開し，再度並列処理階層自動決定手法を適用した場合の (PG, PE) の組合わせが表 2 であり，この指定に基づき DOALL ループを分割した MTG が図 3 である．MTG_0 は (2PG, 2PE) であり，PG0 が MT1・MT3a・MT4a を処理し，PG1 が MT2・MT3b・MT4b を処理する．この場合 PG0 と PG1 では負荷が均等に分散され，処理時間は $10000 + 5000/2 + 5000/2 = 15000[u.t.]$ となる．これは提案手法を用いない並列処理階層自動決定手法を用いた場合に比べ，16.7% の速度向上となる．

5. 性能評価

本章では，提案するインライン展開手法を用いた並列処理階層自動決定手法を OSCAR マルチグレイン並列化コンパイラ[4)~6)] 上に実装し，その性能を 200MHz の Power PC 604e プロセッサを 8 台搭載したミッドレンジ SMP サーバ IBM RS6000 上と，1.1GHz の Power4 プロセッサを搭載した 16 way ハイエンド SMP サーバ IBM pSeries690 regattaH 上で評価した

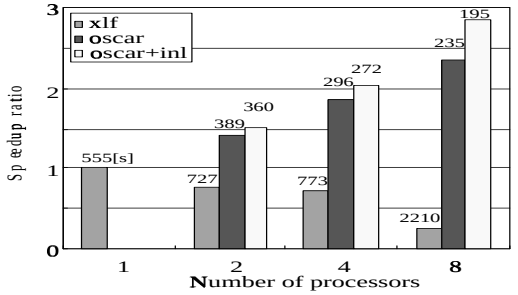


図 6 8 プロセッサまでの turb3d の速度向上率 (RS6000)
Fig. 6 speedup ratio of turb3d up to 8PE (RS6000)

結果について述べる．なお本評価では，提案手法を組み込んだ OSCAR コンパイラを並列化プリプロセッサとして用い，生成された OpenMP API による並列化プログラムを各マシン用のネイティブコンパイラでコンパイルし，実行した．また評価には SPEC95FP のうち，並列性の高いサブルーチンが複数のネストレベルに存在するプログラムである su2cor，mgrid，turb3d を用いた．評価方法は本手法によるインライン展開と階層自動決定手法を用いた場合と，階層自動決定手法のみによる場合の比較である．参考として，XL Fortran コンパイラ の自動ループ並列化性能も示す．

5.1 RS6000 上での性能

RS6000 上での提案手法の性能評価においては OSCAR コンパイラより出力されたプログラムを IBM RS6000 上の IBM XL Fortran Version 7.1 によってコンパイルし，実行した．その際のコンパイルオプションは，提案手法の評価においては XL コンパイラの自動並列化をオフにする “-O3 -qsmp=noauto -qarch=ppc” を用い，XL Fortran コンパイラの自動並列化性能評価においては並列化をオンにする”-O5 -qsmp=auto -qarch=ppc” とした．また逐次性能評価時のオプションは最速実行を可能とする”-O5 -qarch=ppc” を用いた．OSCAR コンパイラ，XL コ

表 3 su2cor の (PG,PE) と並列度
Table 3 (PG, PE) and Parallelism of su2cor

	(PG, PE)	*H_Para*	*Para_inl_ALD*
LOOPS	(1, 8)	125.23	1.00
DO 400	(2, 4)	192.29	3.07
PERM	***	77.42	77.00
MATMAT	***	78.90	78.00
MATADJ	***	82.06	82.00
ADJMAT	***	78.90	78.00
INT2V	(1, 4)	104.54	1.16
INT4V	(1, 4)	161.35	1.16
BESPOL	***	70.74	70.00

表 4 mgrid の (PG,PE) と並列度
Table 4 (PG, PE) and Parallelism of mgrid

	(PG, PE)	*H_Para*	*Para_inl_ALD*
RESID	(8, 1)	11289.83	99.62
RPRJ3	(8, 1)	10836.49	99.60
PSINV	(8, 1)	11045.54	99.61
INTERP	(8, 1)	7784.91	99.78
COMM3	***	44.71	44.56

表 5 turb3d の (PG,PE) と並列度
Table 5 (PG, PE) and Parallelism of turb3d

	(PG, PE)	*H_Para*	*Para_inl_ALD*
GOTO 1001	(8, 1)	1.19	6.00
ENR	***	99.88	2.00
UXW	***	2048.03	64.00
LINAVG	***	4091.22	64.00
MIXAVG	***	2046.59	64.00
LIN	***	4091.22	64.00

ンパイラ共に，su2cor の評価では，配列リネーミングを行ったソースを入力とし，turb3d ではループディストリビューションを行ったソースを入力とした．

図 4，図 5，図 6 に su2cor，mgrid，turb3d の各プロセッサ数に対する速度向上率を示す．これらの表中，横軸がプロセッサ数，縦軸が逐次処理に対する速度向上率を表わし，oscar + inl が提案する選択的インライニング手法を用いたマルチグレイン並列処理，oscar が提案手法を用いないマルチグレイン並列処理，xlf が XL コンパイラのループ並列処理の性能である．図中バー上の数値は実行時間 [s] を表わしている．また，各プログラムの主要な階層の並列度と 8 プロセッサを用いた場合の PG 数・PE 数の組合わせをそれぞれ表 3，表 4，表 5 に示す．表の一番左の列が注目している階層を内包するサブルーチン名またはループ名を表わし，(PG,PE) がその階層のプロセッサ構成すなわちプロセッサグループ数 (PG) とプロセッサエレメント数 (PE) であり *H_Para* はその階層の *Hierarchical_Para* を表わす．

表 3 に示すように su2cor では全実行時間中で約 38%と大きな割合を占めるサブルーチン LOOPS 内の，3 重ループ DO 400 の最内側階層 (ループボディ) の粗粒度並列性が $Para = 2$ と推定されたため，8 プロセッサ利用可能な場合この階層で (2PG，4PE) として 2 プロセッサグループ間で粗粒度並列処理を行っている．このループの内部階層には，下位階層を考慮した並列性 *Hierarchical_Para* が高い SB である PERM，MTAMAT，MATADJ，ADJMAT，INT2V，INT4V が存在し，このうちインライン展開によって上位階層に持ち上げられる並列性 *Para_inl_ALD* が高い PERM，MTAMAT，MATADJ，ADJMAT が LOOPS DO 400 の最内側階層にインライン展開された．一方，INT2V，INT4V は *Para_inl_ALD* が 1.2 弱と小さく，インライン展開しても DO 400 内で増加させることができる並列性が小さいと判断されたため展開の選択はされなかった．これは INT2V，INT4V 内には並列性の高い SB である BESPOL のコール文が複数あるが，これらは逐次処理ループである DO 100 に囲まれておりインライン展開しても LOOPS 内の DO 400 のレベルまで並列性を持ち上げることができないためである．

以上の判定によりインライン展開しない場合 4PE を割り当てられていた PERM，MTAMAT，MATADJ，ADJMAT が，提案手法により (2PG, 4PE) の階層にインライン展開され，これらを処理するプロセッサ数が 4 から 8 へ増えたため，より効果的に並列性が利用されている．実行時間については図 4 に示すように XL Fortran コンパイラの 8 プロセッサまでの最小処理時間 が 4PE のときの 202.64[s] であり，本手法を用いないマルチグレイン並列処理の場合は 110.96 [s] となった．これに対し本手法を用いた場合 98.78 [s] となり，XL コンパイラに比べ 2.05 倍，本手法を用いないマルチグレイン並列処理に比べ 1.12 倍の速度向上が得られた．

またプロセッサ台数効果に関しては図 4 のように，OSCAR コンパイラを用いたマルチグレイン並列処理の場合はプロセッサ数が増えるに従い速度向上が見込めるが，XL コンパイラでは 4 プロセッサを超えると性能が低下している．マルチグレイン並列処理では DO 400 を 2PG で並列処理し，4PE で下位階層の並列性を利用するというように，階層的な並列処理を行なうことが可能である．これに対し XL コンパイラは下位階層に存在する比較的処理コストの小さな DOALL を 8 プロセッサで並列処理しているため同期やスレッド管理のためのオーバーヘッドが大きくなっていると考えられる[9]．

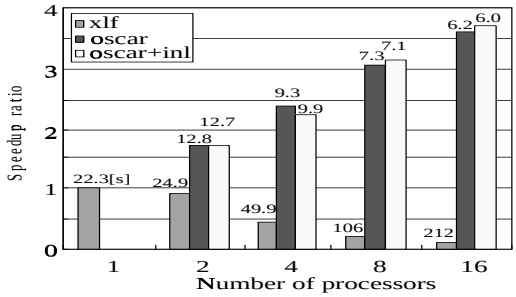


図 7 16 プロセッサまでの su2cor の速度向上率 (regattaH)
Fig. 7 speedup ratio of su2cor up to 16PE (regattaH)

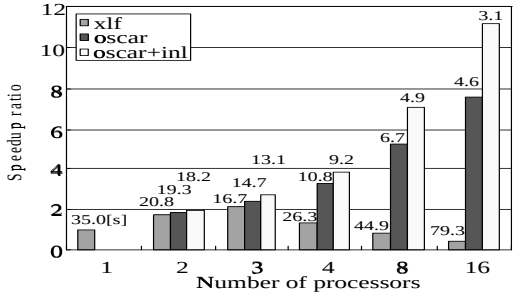


図 8 16 プロセッサまでの mgrid の速度向上率 (regattaH)
Fig. 8 speedup ratio of mgrid up to 16PE (regattaH)

mgrid ではサブルーチン RESID，RPRJ3，PSINV，INTERP の第一階層の $Para\_ALD$ が高く，表 4 のように 8 プロセッサが割り当てられた場合これら第一階層で (8PG，1PE) 構成が選択される．本手法を用いないマルチグレイン並列処理の場合，RESID，RPRJ3，PSINV，INTERP 内で呼ばれるサブルーチン COMM3 には 1 プロセッサしか割り当てられず $Para\_inl\_ALD = 44.56$ である COMM3 の並列性を利用することができないため，図 5 のように 4 プロセッサ以降は速度向上率の増分が小さい．しかし本手法を用いると 2 プロセッサ以上の評価では COMM3 がインライン展開され十分なプロセッサ数で並列処理することが可能であり，8 プロセッサまでほぼスケーラブルな性能向上を示した．最高性能に関しても図 5 に示すように XL コンパイラのみを用いた場合 8 プロセッサまでの最小処理時間は 4PE 使用時の 193.84[s]，本手法を用いないマルチグレイン並列処理の場合が 8PE で 151.17 [s] であるのに対し，本選択的インライン展開手法を用いた場合 84.09 [s] へと処理時間が短縮され，XL コンパイラに比べ 2.30 倍，本手法を用いない場合に比べ 1.79 倍の速度向上が得られた．

turb3d では，サブルーチン TURB3D 内の後方分岐文 (GOTO 1001) によって作られる繰り返しブロック RB がプログラム全体の実行時間のほとんどを占める．この階層は粗粒度並列性が 6 あり，8 プロセッサで turb3d を実行するとこの階層で (8PG, 1PE) という構成が選択され粗粒度タスク並列処理が行われる．しかしこの階層内には表 5 のように $Para\_inl\_ALD$, $Hierarchical\_Para$ が高い SB である ENR, UXW, LINAVG, MIXAVG, LIN が存在するが，1PE しか割り当てられないためこれらの並列性を利用することができない．一方提案手法を用いた場合は，2 プロセッサ以上ではこれら SB がインライン展開され，全てのプロセッサで処理を行なうことが可能となる．このため図 6 が示すようにプロセッサ数が増えるにつれ提案手法を用いない場合との性能差が大きくなり，本手法を用いない場合の最小処理時間は 8PE 使用時の 235.47[s] であるのに対して本手法を用いた場合は 195.18[s] となり，1.21 倍の速度向上となる．XL コンパイラの自動並列化では粗粒度タスク並列性は利用できず turb3d 中の比較的小さなループを並列処理している．しかし，これらのループ並列化のみの効果は小さく，図 6 のように XL コンパイラを用いて 2 プロセッサ以上で並列処理した場合，性能が低下するという結果となった．

5.2 regattaH 上での性能

次に，OSCAR コンパイラを用いて並列化したプログラムを 16 way ハイエンドサーバー regattaH 上で XL Fortran Version 8.1 によってコンパイルし，評価を行った．評価ベンチマークはソースプログラムは RS6000 と同じ su2cor，mgrid，turb3d を用いた．XL Fortran でコンパイルする際のオプションは，OSCAR コンパイラを用いた場合は自動並列化を行わない “-O3 -qsmp=noauto -qarch=pwr4 -qhot” とし，XL Fortran コンパイラの性能評価では mgrid, turb3d に関しては最大レベルの自動並列化を行なう”-O5 -qsmp=auto -qarch=pwr4”，XL コンパイラが-O5 でのコンパイルができなかった su2cor は”-O4 -qsmp=auto -qarch=pwr4’ を用いた．また逐次性能評価のオプションとしては最小処理時間を与える”-O5 -qarch=pwr4”(su2cor は ”-O4 -qarch=pwr4”) を用いた．2，4，8，16プロセッサに対する su2cor，mgrid，turb3d の速度向上率を図 7，8，9 に示す．各図の表記は RS6000 の場合と同一である．

su2cor では LOOPS 内の 3 重ループ DO 400 の最内側階層がインライン展開元の階層として選択され，PERM，MATMAT，MATADJ，ADJMAT がインライン展開された．図 7 を見てみるとプロセッサ数とともに本手法によるインライン展開を行った方がより大きな速度向上を示しており，16プロセッサ使用時の

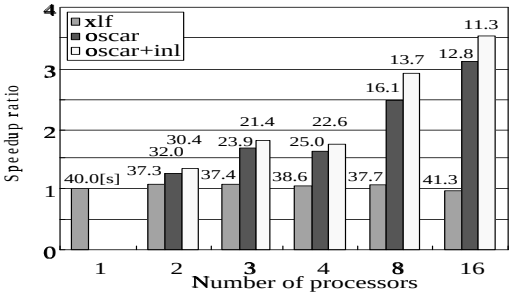


図 9 16 プロセッサまでの turb3d の速度向上率 (regattaH)
Fig. 9 speedup ratio of turb3d up to 16PE (regattaH)

実行時間は本手法を用いないマルチグレイン並列処理が 6.17[s]，用いた場合が 6.00[s] となり，本手法により 3% の速度向上が得られた．また XL コンパイラの自動並列化性能は逐次処理性能を下回るという結果になった．これは regattaH のスレッド生成，スケジューリング，同期などのオーバーヘッドが相対的に大きいためであると考えられる．また 2，4 プロセッサにおける評価では提案手法を用いた場合の方が若干性能が落ちている．これは本手法がマルチグレイン並列性の向上を主目的としキャッシュ最適化を行なっていないため，4 プロセッサ使用時においては提案手法を用いない場合の L3 キャッシュのミス率が 8.83%, 提案手法を用いた場合が 9.50% というようにスケジューリング結果によってはキャッシュミスが増えてしまうためである．

mgrid, turb3d においても XL コンパイラ単体による並列処理性能は同様の傾向を示し，3プロセッサ使用時に mgrid が 16.72[s]，2プロセッサ使用時に turb3d が 37.26[s] という最小の実行時間となった後，プロセッサ数増加に連れて速度向上率は低下している．対してマルチグレイン並列処理では図 8，9 に示すようにプロセッサ数に応じたスケーラブルな性能向上が得られ，本手法を用いないマルチグレイン並列処理では 16 プロセッサ使用時に mgrid で 4.63[s]，turb3d で 12.84[s]，提案手法を用いた場合はプロセッサ数の増加と共にマルチグレイン並列処理との性能差が広がり，16 プロセッサ使用時の処理時間は mgrid で 3.13[s](提案手法を用いない場合の性能に比べ 1.47 倍の性能向上)，turb3d で 11.32[s](1.13 倍の性能向上) となった．

また本評価においては 16 プロセッサまでの実験にとどまっているが，表 3，4，5 の並列度と図 7，8，9 における速度向上率を見る限り更に多くのプロセッサを用いた場合もスケーラブルな性能向上が期待できる．実際にどのくらいのスケーラビリティがあるかは今後評価を行なっていきたい．

6. ま と め

本論文では，階層的粗粒度タスク並列処理において，マルチグレイン並列性を高めるための選択的インライン展開手法を提案した．SPEC95FP ベンチマーク中の 3 本を用いて SMP サーバ IBM RS6000 SP 604e High Node，および IBM pSeries690 regattaH 上で提案手法の評価を行ない，IBM XL Fortran コンパイラに比べ RS6000 上では su2cor で 2.05 倍，mgrid で 2.30 倍，turb3d で 2.84 倍，regatta 上では su2cor で 3.72 倍，mgrid で 5.34 倍，turb3d で 3.29 倍 の速度向上が得られた．また提案する選択的インライン展開を用いない場合のマルチグレイン並列処理に比べ RS6000 上では，su2cor で 1.12 倍，mgrid で 1.79 倍，turb3d で 1.21 倍の速度向上が得られた．また regattaH 上では，su2cor で 1.03 倍，mgrid で 1.47 倍，turb3d で 1.13 倍の速度向上が得られた．これらより提案する選択的インライン展開手法がより効果的なマルチグレイン並列処理を可能とすることが確められた．今後の課題としては，プログラム中のデータの局所性を有効利用するローカライゼーション技術を考慮したインライン展開階層の自動選択が挙げられる．

7. 謝 辞

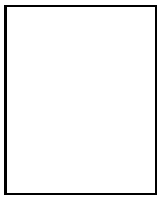
本研究の一部は，ミレニアムプロジェクト IT21 経済産業省/NEDO「アドバンスト並列化コンパイラ」により行われた．

参 考 文 献

1) M.Wolfe: High Performance Compilers for Parallel Computing, *Addison-Wesley Publishing Company* (1996).
2) R.Eigenmann and et al: On the Automatic Parallelization of the Perfect Benchmarks, *IEEE Transaction on parallel and distributed systems, Vol.9* (1998).
3) M.W.Hall and et al: Maximizing Multiprocessor Performance with the SUIF Compiler, *IEEE Computer,* (1996).
4) 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, 情報処理学会誌, Vol.35, No. 4,pp. 513-521 (1994).
5) H.Kasahara and et al: A Multi-grain Parallizing Compilation Scheme on OSCAR, *Proc. 4th Workshop on Language and Compilers for Parallel Computing* (1991).
6) 笠原博徳: 最先端の自動並列化コンパイラ技術, 情報処理, Vol.44 No. 4(通巻 458 号), pp.384-392

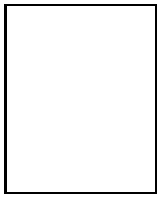
(2003).
7) 本多弘樹, 岩田雅彦, 笠原博徳: Fortran プログラム粗粒度タスク間の並列性検出手法, 電子情報通信学会論文誌, Vol. J73-D-1, No. 12, pp.951-960 (1990).
8) 白子準, 神長浩気, 近藤巧章, 石坂一久, 小幡元樹, 笠原博徳: 並列処理階層自動決定手法を用いた粗粒度タスク並列処理, 情報処理学会研究報告 ARC2002-148-4 (2002).
9) 小幡元樹, 白子準, 神長浩気, 石坂一久, 笠原博徳: マルチグレイン並列処理のための階層的並列性制御法, 情報処理学会論文誌, Vol.44 No. 4, pp.1044-1055 (2003).
10) Zyiyuan, L. and et al.: Interprocedural Analysis for Parallel Computing, *CSRD Technical Report* (1988).
11) H.Paul and et al.: An implementation of Interprocedural Bounded Regular Section Analysis, *IEEE Trans. on Parallel and Distributed Systemss, 2(3):350 360* (1994).
12) M.W.HALL and et al: Detecting Coarse-Grain Parallelism Using an Interprocedural Parallelizing Compiler, *Proc. of Supercomputing '95* (1995).

(平成 15 年 7 月 8 日受付)
(平成 16 年 3 月 8 日採録)



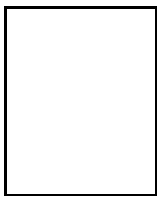
白子　準

昭和 54 年生．平成 14 年早稲田大学理工学部電気電子情報工学科卒業．平成 16 年同大学大学院修士課程修了．平成 16 年同大学大学院博士課程進学，現在に至る．



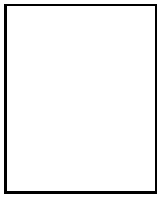
長澤　耕平

昭和 53 年生．平成 13 年早稲田大学理工学部電気電子情報工学科卒業．平成 15 年同大学大学院修士課程修了．現在，富士通株式会社に勤務．在学中はマルチグレイン自動並列化コンパイラに関する研究に従事．



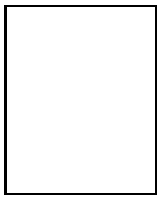
石坂　一久（学生会員）

昭和 51 年生．平成 11 年早稲田大学理工学部電気電子情報工学科卒業．平成 13 年同大学大学院修士課程修了．平成 13 年同大学大学院博士課程進学．平成 14 年同大学同学部助手，現在に至る．



小幡　元樹（正会員）

昭和 48 年生．平成 8 年早稲田大学理工学部電気工学科卒業．平成 10 年同大学大学院修士課程修了．平成 11 年同大学同学部助手．平成 13 年同大学理工学研究科博士課程修了．平成 14 年同大学理工学総合研究センター助手．工学博士．現在は株式会社日立製作所に勤務．在学中はマルチグレイン自動並列化コンパイラに関する研究に従事．



笠原　博徳（正会員）

昭和 32 年生．昭和 55 年早稲田大学理工学部電気工学科卒業．昭和 60 年同大学大学院博士課程修了．工学博士．昭和 58 年同大学同学部助手．昭和 60 年学術振興会特別研究員．昭和 61 年早稲田大学理工学部電気工学科専任講師．昭和 63 年同助教授．平成 9 年同大学教授，現在コンピュータ・ネットワーク工学科．平成元年〜2 年イリノイ大学 Center for Supercomputing Research & Development 客員研究員．昭和 62 年 IFAC World Congress 第一回 Young Author Prize．平成 9 年度情報処理学会坂井記念特別賞受賞．著書「並列処理技術」(コロナ社)．情報処理学会，電子情報通信学会，IEEE などの会員．