OSCAR チップマルチプロセッサ上でのマルチグレイン並列性評価

和田康孝[†] 白子準[†] 石坂一久[†]
木村啓二[†] 笠原博徳[†]

本論文では，コンパイラ協調型 OSCAR チップマルチプロセッサ (OSCAR CMP) 上でのマルチグレイン並列性の評価について述べる．OSCAR CMP は，プログラム中のステートメント間の並列性を利用する近細粒度並列処理，ループイタレーションレベルの並列性を利用する中粒度並列処理，ループやサブルーチン，基本ブロック間の並列性を利用する粗粒度タスク並列処理を階層的に組み合わせて利用するマルチグレイン並列処理を OSCAR マルチグレイン並列化コンパイラと協調して行うことができるように設計されている．このコンパイラとアーキテクチャの協調動作により，OSCAR CMP はチップ上の資源の有効利用およびプログラムの開発効率の向上を可能とする．本論文では，SPEC CFP 95 ベンチマークの，OSCAR CMP 上でのマルチグレイン並列処理性能を評価した結果を報告する．評価の結果，8 プロセッサコアおよび集中共有メモリを 1 チップ上に搭載した OSCAR CMP は，逐次実行に対して，動作周波数が 400MHz であると想定した場合に 2.03〜7.79 倍の性能向上を，動作周波数が 2.8GHz であると想定した場合に 1.89〜7.05 倍の性能向上を得られることが確かめられた．

Evaluation of Multigrain Parallelism on OSCAR Chip Multi Processor

Yasutaka Wada,[†] Jun Shirako,[†] Kazuhisa Ishizaka,[†]
Keiji Kimura[†] and Hironori Kasahara [†]

This paper describes performance of multigrain parallel processing of SPEC CFP 95 on OSCAR Chip Multi Processor(OSCAR CMP). OSCAR multigrain parallelizing compiler, which exploits statement level near-fine grain parallelism, loop iteration level parallelism and coarse grain parallelism hierarchically, allows us to fully control hardware on OSCAR CMP. Also, this cooperation realizes high software productivity and effective use of hardware resources. Performance of multigrain parallel processing of SPEC CFP 95 benchmark programs on OSCAR CMP with 8 processor cores and centralized shared memory were 2.03 to 7.79 times speedup against sequential execution using 400MHz clock cycles for embedded use and 1.89 to 7.05 times speedup against sequential execution using 2.8GHz clock cycles for high-end use.

1. はじめに

近年，1 つのチップ上に複数のプロセッサを集積するチップマルチプロセッサ (CMP) アーキテクチャが種々の分野で導入され始めており，Hydra[1]，Multiscalar[2]，SKY[3]，MP98[4]，Power4[5] など多くのアーキテクチャが提案あるいは商品化されている．しかしながら，チップ上の資源を最大限活用し，CMP アーキテクチャの性能を十分に引き出すためには，プログラム中の並列性を十分に引き出し，チップ上のリソースを適切に利用することのできるコンパイラサポートが必要不可欠である．

これに対し筆者等は，従来から OSCAR マルチグレイン並列化コンパイラおよびこれと協調して動作する OSCAR チップマルチプロセッサ (OSCAR CMP) を提案してきた．このコンパイラ・CMP アーキテクチャの協調は，命令レベルあるいはプログラムの実行文レベルの並列性を利用する近細粒度並列処理に加え，ループイタレーションレベル並列性を利用する中粒度並列処理およびループやサブルーチン，基本ブロック間の並列性を利用する粗粒度タスク並列処理を階層的に組み合わせて利用するマルチグレイン並列処理，データローカリティを有効に利用しプロセッサコア近接メモリを最大限活用するデータローカライゼーション，タスク処理と

† 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,
School of Science and Engineering,
Waseda University

データ転送をオーバーラップして行うデータ転送隠蔽技術など，チップ上資源の最適利用を可能にする．本論文では，特にマルチグレイン並列性の利用に着目し，OSCAR CMP アーキテクチャ上でマルチグレイン並列処理の評価・解析を行った結果について述べる．

以下，2 章でマルチグレイン並列処理について，3 章で OSCAR チップマルチプロセッサアーキテクチャについて，4 章で SPEC CFP 95 を用いた，マルチグレイン並列性利用の解析結果についてそれぞれ述べる．

2. マルチグレイン並列処理

本章では，OSCAR CMP が利用する並列処理手法である，マルチグレイン並列処理について述べる．マルチグレイン並列処理は，ループやサブルーチン，基本ブロック等の粗粒度タスク間の並列性を利用する粗粒度並列処理，ループイタレーションレベルでの並列性を利用する中粒度並列処理，基本ブロック内のステートメント間の並列性を利用する近細粒度並列処理を階層的に組み合わせて，プログラム全域から並列性を抽出して利用する並列処理手法である[6]．このマルチグレイン並列処理は，OSCAR マルチグレイン並列化コンパイラに実装されている．

2.1 粗粒度タスク並列処理

粗粒度タスク並列処理では，コンパイラはまず対象とするプログラムを擬似代入文ブロック (BPA)，繰り返しブロック (RB)，サブルーチンブロック (SB) の 3 種類の粗粒度タスク (マクロタスク (MT)) に分割する．MT 生成後，MT 間のコントロールフローおよびデータ依存を解析し，マクロフローグラフ (MFG) を作成し，さらに最早実行可能条件解析によって MFG から MT 間の並列性を抽出してマクロタスクグラフ (MTG) を生成する[7][8]．MFG および MTG の例を図 1 に示す．その後，コンパイラは各 MT を 1 つ以上のプロセッサエレメント (PE) をグループとしたプロセッサグループ (PG) に割り当てる．このとき，MTG 内に条件分岐等がなければ，プロセッサ間の同期やデータ通信などのオーバーヘッドを最小化するために静的に MT を割り当てる (スタティックスケジューリング)．逆に条件分岐等によって実行時不確定性が存在する場合には，MT のコードに加えて，MT を PG に割り当てるためのスケジューラのコードも合わせて生成し，実行時に MT を PG に割り当てるようにする (ダイナミックスケジューリング)．

さらに，MTG 内の SB や RB 内に粗粒度並列性が存在する場合，その SB や RB 内部でさらに MTG を生成し，階層的に粗粒度タスク並列処理を適用する．

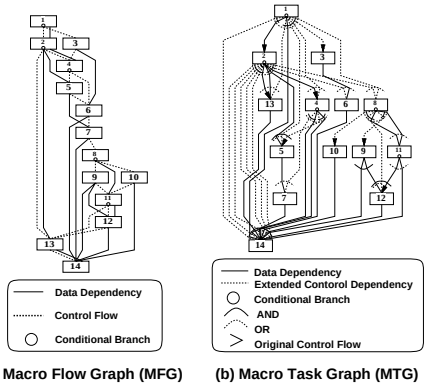
2.2 中粒度並列処理

PG に割り当てられた RB が DOALL あるいはリダクションループなどの並列化可能ループであった場合，この RB は PG 内の PE を用いてイタレーションレベルの並列処理が適用される．

2.3 近細粒度並列処理

PG に割り当てられた MT が BPA あるいは中粒度並列処理が適用不可能な RB であった場合，BPA 全体，あるいは RB のループボディに対して近細粒度並列処理[9] が適用される．

近細粒度並列処理においては，BPA 内部のステートメントあるいは複数のステートメントからなる擬似代入文を 1 つの近細粒度タスクとして定義する．コンパイラはこれらのタスク間のデータ依存を解析してタスクグラフを作成し，このタスクグラフ上の各タスクのコストおよびタスク間のデータ転送コストを考慮して，実行時間を最小化できるように PG 内の PE に対して静的にスケジューリングする．



(a) Macro Flow Graph (MFG)　(b) Macro Task Graph (MTG)

図 1 マクロフローグラフ (MFG)，マクロタスクグラフ (MTG) の例

OSCAR マルチグレイン並列化コンパイラにおける近細粒度タスクの PE へのスケジューリングでは，スケジューリング手法として，データ転送オーバーヘッドを考慮し実行時間を最小化するヒューリスティックアルゴリズムである CP/DT/MISF 法，CP/ETF/MISF 法，ETF/CP 法，および DT/CP 法の 4 つの手法の中から最良のスケジュール結果を与えるものが選択される．

3. OSCAR チップマルチプロセッサアーキテクチャ

本章では，第 2 章で述べたマルチグレイン並列処理を効率よく行うために設計された OSCAR チップマルチプロセッサ (OSCAR CMP) アーキテクチャについて述べる．

OSCAR CMP は，簡素なプロセッサコア，ローカルプログラムメモリ (LPM)，ローカルデータメモリ (LDM)，2 ポート構成の分散共有メモリ (DSM) およびデータ転送ユニット (DTU) をもつプロセッサエレメント (PE) をチップ上に複数集積し，これらと集中共有メモリ (CSM) を複数バスやクロスバ等の結合網で接続したものである．

LPM は各 PE が実行するプログラムを格納し，LDM はアプリケーションプログラム中においてコンパイラが検出あるいは分割配置により割り当てたプロセッサプライベートデータを格納する．DSM は上述のように 2 ポート構成となっており，他の PE からも同時にアクセスが可能である．DSM は，粗粒度タスク並列処理や近細粒度並列処理におけるデータ・同期フラグの授受に利用される．DTU はプロセッサの処理と非同期に PE 間，PE-CSM 間のデータ転送を行うことが可能であり，データ転送とタスク処理のオーバーラップのために利用する．また，簡素なプロセッサコアを用いることにより，近細粒度並列処理において，コンパイラが各 PE に対してスケジューリングを行った結果をプログラム実行時に忠実に再現することができる．

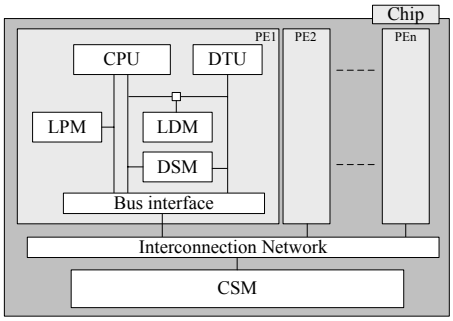


図 2 OSCAR チップマルチプロセッサアーキテクチャ

OSCAR CMP が備えるこれらのハードウェアをコンパイラが適切に制御して利用することで，効率の良いマルチグレイン並列処理を行うことができるようになっている．[10] なお，本論文の評価では，CSM もチップ上に集積されているものとする．OSCAR CMP のアーキテクチャ図を図 2 に示す．

4. 性能評価

本章では，OSCAR CMP 上で SPEC CFP 95 を用いてマルチグレイン並列性を評価した結果について述べる．

4.1 評価環境

本評価では，OSCAR CMP 上に PE を 1〜8 基搭載するものとした．CSM は複数のバンクからなるものとし，PE 間結合網は複数バスを用いた．本評価における PE 間結合網は 3 本バス，CSM の構成は 4 バンク構成とした．

プロセッサ動作周波数としては，組み込み向け低消費電力プロセッサを想定した 400MHz，およびハイエンドプロセッサを想定した 2.8GHz の条件で評価を行った．PE 内の各種メモリおよび CSM の容量およびアクセスコスト，PE 間結合網の遅延等については表 1 に示すとおりである．なお，メモリアクセスコストの値は 90nm プロセスを想定し，CACTI[11] を用いて算出した値を基にしている．

本評価では，各 PE の持つプロセッサコアは，SPARC V9 規格に準拠したプロセッサである Sun Microsystems 社の UltraSPARC-II のパイプライン構成をベースとし，バリア同期機構等用の特殊レジスタや特殊レジスタを操作するための命令を付加したプロセッサで，整数演算ユニット (IEU) を 1 本，ロードストアユニット (LSU) を 1 本，浮動小数点ユニット (FPU) を 1 本持つシングルイシューのシンプルな構成とした．

評価には，上記の構成を持つ OSCAR CMP を精密に再現するシミュレータを用いた．

表 1 OSCAR CMP における各メモリのアクセスコスト

想定動作周波数	400MHz	2.8GHz
DSM(32kB, 2Port)	1 クロック	2 クロック
LDM(128kB)	1 クロック	2 クロック
CSM(3MB)	2 クロック	8 クロック
ネットワーク遅延 (調停を含む)	2 クロック	10 クロック

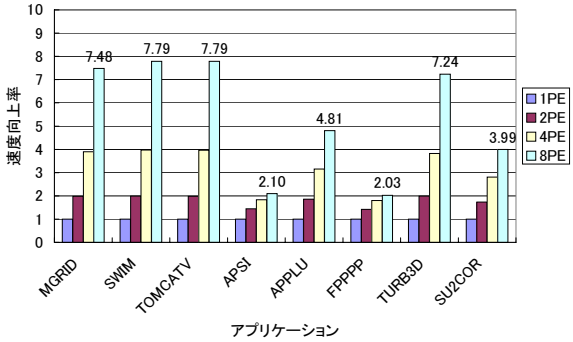


図 3 OSCAR CMP 上でのマルチグレイン並列処理の性能評価結果 (400MHz)

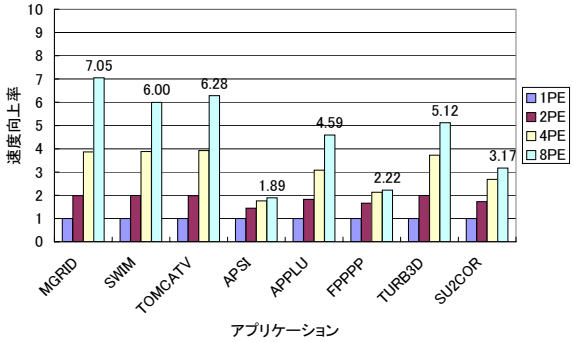


図 4 OSCAR CMP 上でのマルチグレイン並列処理の性能評価結果 (2.8GHz)

また，本評価では SPEC CFP 95 に含まれる APPLU, APSI, FPPPP, MGRID, SU2COR, SWIM, TOMCATV, TURB3D の 8 本の FORTRN77 プログラムを用いて評価を行った．なお，評価時間を短縮するためプログラム内のパラメータや入力データを修正している．これらのプログラムに OSCAR マルチグレイン並列化コンパイラによるマルチグレイン並列処理を適用し，実行バイナリを生成した．ただし，並列性の評価が主な目的であるため，データローカリティ最適化技術や DTU を用いたデータ転送隠蔽技術などは利用していない．

4.2 評価結果

動作周波数 400MHz を想定した場合の評価結果を図 3 に，2.8GHz を想定した場合については図 4 に示す．図 3 および図 4 において，横軸は評価に用いたアプリケーション名，縦軸は逐次処理に対する並列処理時の速度向上率を表している．また，4 本のバーは各々左から 1PE，2PE，4PE，8PE を使用した時の逐次実行に対する速度向上率を示している．

以下，4.2.1 節で主に中粒度並列性を利用したアプリケーションについて，4.2.2 節で並列性の低いアプリケーションについて，4.2.3 節で主に近細粒度並列性を利用したアプリケーションについて，4.2.4 節で粗粒度並列性を利用したアプリケーションについてそれぞれ述べる．

4.2.1 中粒度並列性を持つアプリケーション

MGRID，SWIM，TOMCATV，TURB3D はループ並列性に富むアプリケーションであり，図 3 および図 4 から，逐次実行時に対し，400MHz 時に MGRID で最大 7.48 倍，SWIM で最大 7.79 倍，TOMCATV で最大 7.79 倍，TURB3D で最大 7.24 倍，2.8GHz 時に MGRID で最大 7.05 倍，SWIM で最大 6.00 倍，TOMCATV で最大 6.28 倍，TURB3D で最大 5.12 倍と，PE 数に応じたスケーラブルな性能向上を得ることができている．ただし，TOMCATV の評価結果については，ファイル入出力部分を除いた値となっている．

TURB3D では粗粒度並列性も併せて利用しているが，これについては 4.2.4 節で述べる．

4.2.2 並列性の低いアプリケーション

APSI は近細粒度並列性，粗粒度並列性ともに乏しく，主にループ並列性を利用しているアプリケーションである．しかしながら，400MHz 時に最大 2.10 倍，2.8GHz 時に最大 1.89 倍と他のアプリケーションに比べて速度向上率が低い．これは，並列化可能ループを実行している時間が実行時間全体に占める割合は大きいが，それらのループの多くは回転数が小さく，かつループボディも小さいためである．例えば，本評価で用いた入力データを用いて 400MHz で逐次実行した場合，並列化可能ループがアプリケーション全体の処理時間に占める割合はおよそ 83%であるが，そのうち回転数 1 のものが全体の約 15%，回転数 2 のものが全体の約 6%，回転数 8 のものが全体の約 13%を占めている．このことから，APSI では中粒度並列処理を利用しても得られる並列性は小さく，400MHz で最大 2.10 倍，2.8GHz で最大 1.89 倍と言う本性能評価結果は，アプリケーションの並列性を考えると十分な性能であると言える．

4.2.3 近細粒度並列性を持つアプリケーション

a) APPLU

APPLU は，ほぼ同様の構成を持つサブルーチン BLTS および BUTS 内にループを持ち，これが全体の実行時間の内，大きな割合を占めている．

これらのループはパイプライン並列化が可能であることが知られている．しかしながら，これらのループの最内側ループは回転数が少なく，しかも固定なので，ループアンローリングを適用することにより 1 つの大きな基本ブロックを持つ完全ネストループにリストラクチャリング可能である．本評価では，リストラクチャリング後の BLTS，BUTS に近細粒度並列処理を，その他の部分にループ並列処理を適用するマルチグレイン並列処理を適用して評価を行った．

図 5 に，サブルーチン BLTS および BUTS に対して中粒度並列処理を適用した場合と，ループアンローリングおよび近細粒度並列処理によるマルチグレイン並列処理を適用した場合の APPLU の速度向上率を示す．なお，これらのサブルーチン以外の部分については両者とも同様の並列化を適用している．図 5 より，中粒度並列処理適用時に 400MHz 動作で最大 2.92 倍，2.8GHz 動作で最大 2.82 倍の性能向上に留まっているのに対し，近細粒度並列処理適用時には 400MHz 動作で最大 4.81 倍，2.8GHz 動作で最大 4.59 倍と大きな性能向上を得ることができている．

次に，OSCAR CMP アーキテクチャが近細粒度並列処理においてコンパイラの引き出した並列性を十分に利用できているかについて述べる．まずある基本ブロックにおいてコンパイラの引き出した近細粒度並列性を式 (1) を用いて定義する．式 (1) において，$EffectivePara(i)$ は基本ブロック $Block_i$ においてコンパイラが抽出した並列性，Seq_i は注目ブロックの逐次実行時のコスト，$Sche_i$ はコンパイラが各 PE に対してスケジューリングを行った際のスケジュール長である．この $EffectivePara$ は，コンパイラが想定したとおりにアーキテクチャが動作した場合に得られる速度向上率の期待値である．

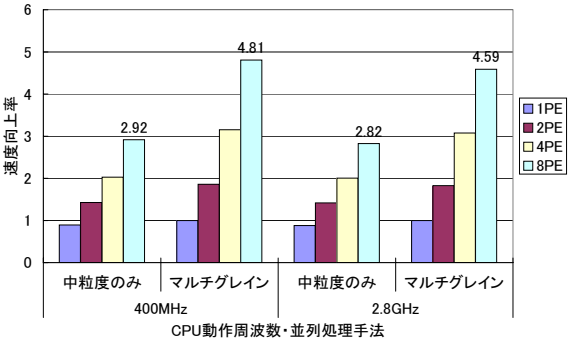


図 5 並列処理手法による APPLU の速度向上率の比較

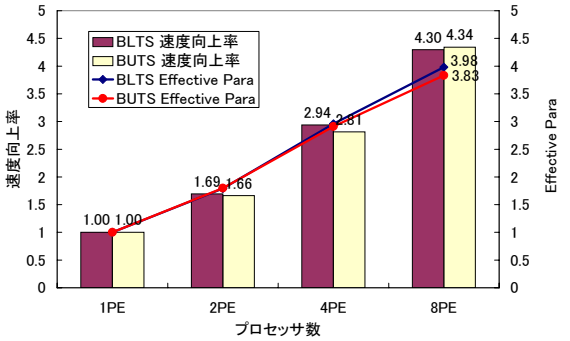


図 6 APPLU における速度向上率とコンパイラの引き出した近細粒度並列性の比較 (2.8GHz)

$$EffectivePara(i) = \frac{Seq_i}{Sche_i} \quad (1)$$

図 6 に APPLU のサブルーチン BLTS および BUTS に対して，ループアンローリングおよび近細粒度並列処理を適用した場合の速度向上率と $EffectivePara$ を示す．図 6 中の横軸はプロセッサ数，縦軸は逐次実行に対する速度向上率および $EffectivePara$ である．図 6 を見ると，それぞれのサブルーチンの速度向上率はほぼ $EffectivePara$ と同様の推移を示している．このことから，OSCAR CMP アーキテクチャは近細粒度並列処理においてコンパイラの引き出した並列性を十分に利用できていると言える．

b) FPPPP

FPPPP はループ並列性，粗粒度並列性が共にほとんど存在しないが，近細粒度並列性の高いアプリケーションであり，プログラムのほぼ全域に対して近細粒度並列処理を適用す

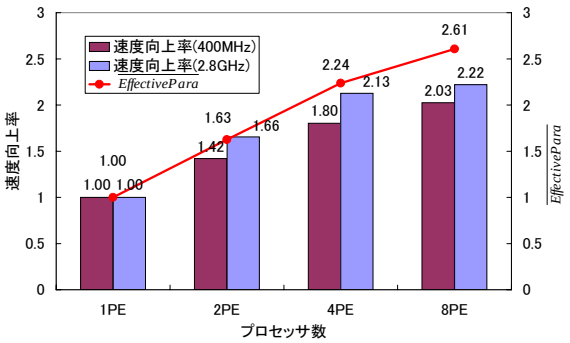


図 7 FPPPP の速度向上率とコンパイラの引き出した並列性の比較

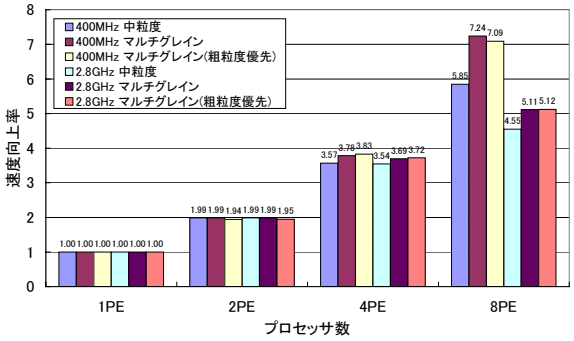


図 8 並列処理手法による TURB3D の速度向上率の比較

ることで，逐次実行に対し 400MHz 動作時に最大 2.03 倍，2.8GHz 動作時に最大 2.22 倍の性能向上を得ている．

ここで，式 (1) を拡張し，プログラム全体の近細粒度並列性を $\overline{EffectivePara}$ として式 (2) を用いて定義する．式 (2) において，$count_i$ は基本ブロック $Block_i$ の実行回数を表し，他の値は式 (1) と同様である．

$$\overline{EffectivePara} = \frac{\sum_{i=1}^{n} Seq_i \times count_i}{\sum_{i=1}^{n} Sche_i \times count_i} \qquad (2)$$

実際には，この $\overline{EffectivePara}$ は近細粒度並列処理が適用されたブロックのみを対象としているため，近細粒度並列性以外の並列性を利用して速度向上を得ているプログラムにおいては，この値だけでプログラム全体の並列性を表すことはできない．しかし，FPPPP では上述のように速度向上のほぼ全てが近細粒度並列処理によるため，この $\overline{EffectivePara}$ をプログラム全体の速度向上の期待値とみなすことができる．図 7 に FPPPP の速度向上率と $\overline{EffectivePara}$ を示す．図の横軸はプロセッサ数，縦軸は逐次実行に対する速度向上率および $\overline{EffectivePara}$ である．図 7 より，$\overline{EffectivePara}$ は同期等のコストを考慮していないため速度向上率の方が若干低い値を示しているが，速度向上率と $\overline{EffectivePara}$ はほぼ同様の推移を示していることがわかる．すなわち，APPLU のように並列化可能ループをアンローリングしたステートメント間の依存が少ない基本ブロックだけではなく，複雑な依存関係を持った基本ブロックに対しても，OSCAR CMP アーキテクチャはコンパイラの引き出した並列性を反映した性能を示しているといえる．

4.2.4 粗粒度並列性を持つアプリケーション

a) TURB3D

TURB3D は実行時間のほとんどをサブルーチン TURB3D が占め，サブルーチン TURB3D はその内部にサブルーチン XYFFT，ZFFT を呼び出すコストの大きなループを複数持つ．これらのループは並列化可能であり，ループディストリビューションを施すことで粗粒度並列性も抽出できる．また，サブルーチン TURB3D 内にはこれらのループ以外にも粗粒度並列性を持つ箇所が存在する．本評価では，このサブルーチン TURB3D に対して，

1. 粗粒度並列性を用いず，ループ並列性を利用した場合 (中粒度)
2. 上述のループ群に関しては中粒度並列処理を適用し，それ以外の箇所は粗粒度並列性を利用した場合 (マルチグレイン)
3. 上述のループ群に対してループディストリビューションを施し，粗粒並列性を優先的に利用した場合 (マルチグレイン (粗粒度優先))

の 3 種類の並列化を行った場合について評価を行った．2. ではループ並列性を優先的に利用しているのに対し，3. では，ループディストリビューション後の個々のループが粗粒度タスクとして PG に割り当てられ，PG 内の PE を用いて中粒度並列処理を行う階層的な並列処理が行われる形になる．なお，サブルーチン TURB3D 以外の箇所については同様の並列化を適用している．これら 3 種類の並列化を行った場合の速度向上率を図 8 に示す．図 8 において，横軸はプロセッサ数，縦軸は逐次実行に対する速度向上率である．

図 8 では，プロセッサ数が 4PE 以下の少ない場合であれば三者ともほぼ同等の性能を示している．しかしながら，プロセッサ数が増えた場合，中粒度並列処理のみでは速度向上が得られていないのに対し，粗粒度並列性を利用した場合には，スケーラブルな性能向上を維持していることがわかる．また，2. と 3. を比べると，PE 数が 8 までの範囲であればほぼ同等の性能を示している．これは，上述のループ群のコストが PE 数と比較して十分大きく，ループ制御やダイナミックスケジューリングのオーバーヘッドが相対的に小さく抑えられているためである．

b) SU2COR

SU2COR はサブルーチン LOOPS 内に粗粒度並列性を持つループが存在し，そのループ内の粗粒度タスクが内部にループ並列性を併せ持つアプリケーションである．ただし，ループ並列性を持つ箇所がネストの深い部分に存在するため，コンパイラは同期等のオーバーヘッドを考慮してより階層の浅い部分に存在する粗粒度並列性を優先して利用する選択をする．しかし，チップマルチプロセッサのような同期のオーバーヘッドを低く抑えることのできるアーキテクチャでは，ネストの深い箇所のループ並列性を利用した場合でも性能向上を得ることができる可能性がある．よって本評価では，サブルーチン LOOPS において粗粒度並列性を優先して利用した場合に加え，粗粒度並列性を用いずにループ並列性を利用した場合についても評価を行った．粗粒度並列性を利用した場合は，粗粒度タスクが PG に割り当てられ，PG 内の PE を用いて各粗粒度タスク内で中粒度並列処理を行う階層

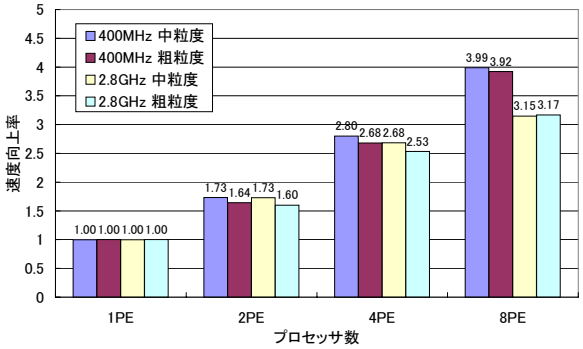


図 9 並列処理手法による SU2COR の速度向上率の比較

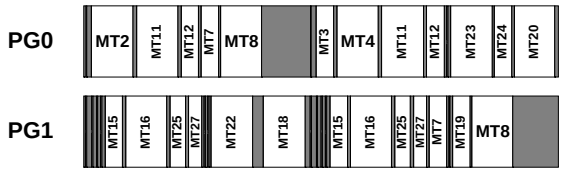


図 10 マルチグレイン並列処理時の SU2COR の実行トレースの例

的マルチグレイン並列処理が適用される．

SU2COR における両者の性能の比較を図 9 に示す．なお，図中の横軸はプロセッサ数，縦軸は逐次実行に対する速度向上率である．図 9 より，逐次実行に対し 400MHz 動作時に最大 3.99 倍，2.8GHz 動作時に最大 3.17 倍の性能向上を得ることができている．また，粗粒度並列性を利用して階層的マルチグレイン並列処理を行った場合と，粗粒度並列性を用いずループ並列性を利用した場合でほぼ同等の性能を示している．

ここで，階層的マルチグレイン並列処理を適用した際の実行トレースの例を図 10 に示す．図 10 から，粗粒度並列性を利用した場合には，MT 間の依存関係によって PG に処理が割り当てられていない時間が発生している．すなわち，タスクの処理時間，依存関係ををきちんと考慮し，タスク分割等の技術を利用してこれらの処理未割り当て時間を軽減すれば，単純にループ並列性を利用した場合に比べて大きな性能向上を得ることが可能である．

さらに，今後集積可能な PE 数が増加していくことを考えると，単純にループ並列性を用いるだけではループ回転数以上の並列性を得ることができないため，PE 数に応じた性能向上を得ることはできない．しかしながら，粗粒度並列性を併用し階層的にマルチグレイン並列処理を適用することでより大きな並列性を引き出すことができ，多くのプロセッサを有効に利用することができると考えられる．

5. ま と め

本論文では，ソフトウェア協調動作型 OSCAR チップマルチプロセッサ (OSCAR CMP) 上でのマルチグレイン並列処理性能について述べた．SPEC CFP 95 ベンチマークを用いた性能評価では，簡素なシングルイシューのプロセッサコアを持つプロセッサエレメント (PE)8 基，および集中共有メモリを 1 チップ上に搭載した OSCAR CMP は，逐次実行に対して，動作周波数が 400MHz であると想定した場合に 2.03〜7.79 倍の性能向上を，動作周波数が 2.8GHz であると想定した場合に 1.89〜7.05 倍の性能向上を得られることが確かめられた．また，各アプリケーションの並列性の特徴を考慮しつつ得られた性能を解析した結果，コンパイラはプログラムの各部分から適切な粒度の並列性を抽出し，アーキテクチャがそれらの並列性を十分に利用できていることが確認できた．

今後の課題としては，さらに PE 数が増加した場合や HW コストの制限がある場合に問題となる，PE 間結合網などのリソース競合に対する対応，コンパイラと協調した低消費電力化などが挙げられる．

謝辞

本研究の一部は，STARC「自動並列化コンパイラ協調型シングルチップマルチプロセッサの研究」及び早稲田大学理工総研プロジェクト研究「自動並列化コンパイラ協調型チップマルチプロセッサ」により行われた．本論文作成にあたり有益なコメントをいただいた，宮田操氏 (STARC)，高橋宏政氏 (富士通研)，高山秀一氏 (松下)，安川英樹氏 (東芝)，倉田隆弘氏 (ソニー) に感謝致します．

参 考 文 献

1) Hammond, L., Hubbert, B., Siu, M., Prabhu, M. K., Chen, M. and Olukotun, K.: The Stanford HYDRA CMP, *IEEE MICRO*, Vol. 19, No. 2 (1999).
2) Sohi, G., Breach, S. and Vijaykumar, T.: Multiscalar Processors, *Proc. 22th International Symposium on Computer Architecture (ISCA-22)* (1995).
3) 小林, 岩田, 安藤, 島田: 非数値計算プログラムのスレッド間命令レベル並列を利用するプロセッサ・アーキテクチャSKY, *JSPP'98*, pp. 87–94 (1998).
4) Edahiro, M., Matsushita, S., Yamashita, M. and Nishi, N.: A Single-Chip Multiprocessor for Smart Terminals, *IEEE MICRO* (2000).
5) Tendler, J. M., Dodson, S., Fields, S., Le, H. and Sinharoy, B.: *POWER4 System Microarchitecture* (2001).
6) Kasahara, Honda and Narita: A Multigrain Parallelizing Compilation Scheme for OSCAR, *Proc.4th Workshop on Lang. And Compilers for Parallel Computing* (1991).
7) 本田, 岩田, 笠原: Fortran プログラム粗粒度タスク間の並列性検出法, 信学論 (D-I), Vol. J73-D-I, No. 12, pp. 951–960 (1990).
8) 笠原, 合田, 吉田, 岡本, 本多: Fortran マクロデータフロー処理のマクロタスク生成手法, 信学論, Vol. J75-D-I, No. 8, pp. 511–525 (1992).
9) 木村, 尾形, 岡本, 笠原: シングルチップマルチプロセッサ上での近細粒度並列処理, 情報処理学会論文誌, Vol. 40, No. 5 (1999).
10) Kimura, K., Kodaka, T., Obata, M. and Kasahara, H.: Multigrain Parallel Processing on Compiler Cooperative OSCAR Chip Multiprocessor Architecture, *The IEICE Transactions on Electronics, Special Issue on High-Performance and Low-Power System LSIs and Related Technologies*.
11) Wilton, S. and Jouppi, N.: CACTI: An enhanced cache access and cycle time model, *IEEE Journal of Solid-State Circuits*, Vol. 31, No. 5, pp. 677–688 (1996).