SMP サーバ及び組込み用マルチコア上での OSCAR マルチグレイン自動並列化コンパイラの性能

白子 準[†] 田川友博[†] 三浦 剛[†]
宮本孝道[†] 中野啓史[†]
木村啓二[†] 笠原博徳[†]

半導体集積度向上に伴うスケーラブルな性能向上，低消費電力，価格性能を達成するためにマルチコアプロセッサが大きな注目を集めている．このようなマルチコアプロセッサの性能を最大限に引き出し，ソフトウェア/ハードウェア開発期間を短縮するためには自動並列化コンパイラが重要な役目を果たす．本論文ではループ並列処理に加え，粗粒度タスク並列処理・近細粒度並列処理によりプログラム全域にわたる並列化を行う OSCAR マルチグレイン自動並列化コンパイラを用いた，最新 SMP サーバ及び組込み組込み用マルチコアプロセッサ上での性能評価について述べる．OSCAR コンパイラではプログラム中の各部分に対する適切な処理プロセッサ数と並列処理手法の決定，複数のループや粗粒度タスク間にまたがる広域的なキャッシュメモリ最適化技術が実現されている．SPEC CFP95 ベンチマーク全 10 本と CFP2000 ベンチマーク 4 本を用いた性能評価において，OSCAR コンパイラは IBM p5 550Q Power5+ 8 プロセッササーバ上で IBM XL Fortran コンパイラ version 10.1 の自動並列化性能に比べ平均 2.74 倍，IBM pSeries690 Power4 24 プロセッササーバ上で IBM XL Fortran コンパイラ version 8.1 の自動並列化性能に比べ平均 4.82 倍の性能向上が得られた．また NEC/ARM MPCore ARMv6 4 プロセッサ集積組込み用マルチコアにおいて，OpenMP API の一部機能をサポートすることで OSCAR コンパイラによる自動並列化を実現した．組込み用途を考慮しデータセットを縮小した SPEC CFP95 を用いた評価において，逐次処理に比べ tomcatv で 4.08 倍，swim で 3.90 倍，su2cor で 2.21 倍，hydro2d で 3.53 倍，mgrid で 3.85 倍，applu で 3.62 倍，turb3d で 3.20 倍の性能向上が得られた．

Performance of OSCAR Multigrain Parallelizing Compiler on SMP Servers and Embedded Multicore

Jun shirako,[†] Tomohiro Tagawa,[†] Tsuyoshi Miura,[†]
Takamichi Miyamoto,[†] Hirofumi Nakano,[†] Keiji Kimura[†]
and Hironori Kasahara [†]

Currently, multiprocessor systems, especially multicore processors, are attracting much attention for performance, low power consumption and short hardware/software development period. To take the full advantage of multiprocessor systems, parallelizing compilers serve important roles. This paper describes the execution performance of OSCAR multigrain parallelizing compiler using coarse grain task parallelization and near fine grain parallelization in addition to loop parallelization, on the latest SMP servers and a SMP embedded multicore. The OSCAR compiler has realized the automatic determination of parallelizing layer, which decides the suitable number of processors and parallelizing technique for each nested part of the program, and global cache memory optimization over loops and coarse grain tasks. In the performance evaluation using 10 SPEC CFP95 benchmark programs and 4 SPEC CFP2000, OSCAR compiler gave us 2.74 times speedup compared with IBM XL Fortran compiler 10.1 on IBM p5 550Q Power5+ 8 processors server, 4.82 times speedup compared with IBM XL Fortran compiler 8.1 on IBM pSeries690 Power4 24 processors server. OSCAR compiler can be also applied for NEC/ARM MPCore ARMv6 4 processors low power embedded multicore, using subset of OpenMP libraries and g77 compiler. In the evaluation using SPEC CFP95 benchmarks with reduced data sets, OSCAR compiler achieved 4.08 times speedup for tomcatv, 3.90 times speedup for swim, 2.21 times speedup for su2cor, 3.53 times speedup for hydro2d, 3.85 times speedup for mgrid, 3.62 times speedup for applu and 3.20 times speedup for turb3d against the sequential execution.

1. はじめに

高実効性能，低消費電力性，高価格性能およびソフトウェア/ハードウェア開発期間短縮といった観点から，マルチコアプロセッサが大きな注目を集めている．例えば，ソニー/IBM/東芝の Cell[1]，NEC/ARM の MPCore，MP211[2]，富士通 FR-V[3]，パナソニック Uniphier，ルネサステクノロジ SH-X3 といった 情報家電向け組込み用マルチコアや，PC，サーバ向けのインテル Dual コア Xeon[4] や Core 2 Duo，AMD の Dual/Quad コア Opteron[5]，ワークステーション，ハイエンドサーバ用に開発された Sun SPARC T1，T2，そして IBM Power4, 5, 5+[6] などが挙げられる．

† 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

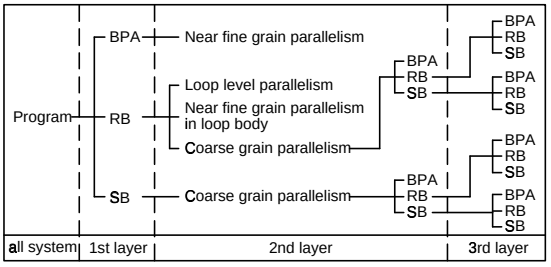


図 1 階層的マクロタスク定義

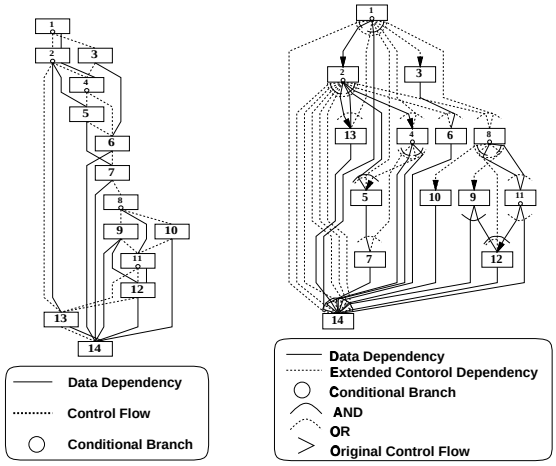
これらマルチコアプロセッサを用いたシステムにおいて高い実効性能を実現するためには，実行するプログラムに対する適切なグレイン (粒度) での並列性抽出，メモリウォール問題を克服するためのキャッシュ及びローカルメモリの最適化，DMA を用いたプロセッサ間のデータ転送の最小化が必須となる．これらの実現のため従来より自動並列化コンパイラの研究が行われており[7)~9)]，ループ並列化技術は大きな進歩を遂げた．しかしながら現在ではループ並列化手法は成熟期に至っており，今後マルチコアシステム上での大幅な性能向上を達成するためにループ並列性以外の並列性を利用する並列化手法が必要とされている．マルチレベルの並列性を利用するコンパイラとしては NANOS コンパイラ[10)]，PROMIS コンパイラ[11)]，そして OSCAR コンパイラ[12)~14)] が挙げられる．OSCAR マルチグレイン自動並列化コンパイラでは，プログラム中の粗粒度タスク並列処理，ループレベル並列処理，近細粒度並列処理を組み合わせたマルチグレイン並列処理を実現している．また OSCAR コンパイラは抽出したマルチグレイン並列性に応じ，プログラムの各部分の並列性に見合った適切なプロセッサの割当てや複数のループ (すなわち粗粒度タスク) 間にまたがる広域的なキャッシュメモリ最適化も実現している．本論文では OSCAR コンパイラにおいて用いられているこれらの技術と，最新 SMP サーバ及び組込み用マルチコアプロセッサ上での性能評価について述べる．

2. マルチグレイン並列処理

本章では，マルチグレイン並列処理における粗粒度タスク並列処理について述べる．粗粒度タスク並列処理とは逐次プログラムを階層的に粗粒度タスク分割し，生成された粗粒度タスクすなわちマクロタスクをプロセッサエレメント (PE)，もしくはプロセッサグループ (PG) 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[15)]．

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である BPA，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，サブルーチンブロック SB の 3 種類のマクロタスク MT[14)] 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，その内部をさらにマクロタスク分割し階層的なマクロタスク構造を生成する (図 1)．



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

図 2 粗粒度タスク並列性の抽出

2.2 粗粒度タスク並列性抽出

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フローを解析し，図 2 (a) に示すようなマクロタスク間のデータと制御のフローを表すマクロフローグラフ[12),14)] を生成する．次に，階層的に生成されたマクロフローグラフに対し最早実行可能条件解析[12),14)] を適用し，階層的なマクロタスクグラフ MTG[12),14)] を生成する (図 2 (b))．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よい条件であり，マクロタスクグラフが粗粒度タスク並列性を表す．

2.3 階層的なプロセッサグルーピング

階層的なマクロタスクグラフを効果的に処理するため，プロセッサのグルーピングを行う．複数のプロセッサエレメント PE をソフトウェア的にグループ化し，プロセッサグループ PG を定義する．この PG がマクロタスクを処理する単位であり，SB や RB など内部にマクロタスクグラフが存在する場合はプロセッサグループ内の PE を更にグルーピングする．図 3 に階層的なグルーピングの例を示す．図中，プログラム全体で 8 プロセッサ利用可能であるとする．第 1 階層 (1st layer) では 4 つの PE を持った 2 つの PG を定義し，これを (2PG, 4PE) の構成と表記する．第 1 階層ではこの 2 つの PG を用い 2 並列の粗粒度タスク並列処理を行ない，更に 4 PE を割り当てられた第 2 階層 (2nd layer) ではそれぞれ (4PG,1PE) で 4 並列，(2PG, 2PE) で 2 並列という粗粒度タスク並列処理を行っている．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マクロタスクグラフの並列性を有効に利用することが可能である．

2.4 並列処理階層自動決定手法

抽出された各階層の粗粒度タスク並列性をプロセッサに効率よく割当てることが，マルチグレイン並列処理の性能向上において重要である．OSCAR コンパイラに実装されている並列処理階層自動決定手法[15),16)] では，推定した各階層のマクロタスクグラフの並列度をもとにそれぞれの階層に対して適切な並列処理手法を選択し，PG

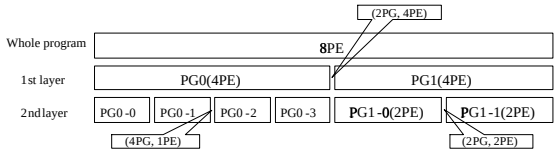


図 3 プロセッサグループ・プロセッサエレメントの階層的定義

数・PE 数の決定を行う．これにより，抽出されたマルチグレイン並列性を効果的に利用している．さらに各マクロタスクの並列性の最大値を推定し，そのマクロタスクを処理するのに必要十分なプロセッサ数を推定する．これにより過剰と判断されたプロセッサにはオーバーヘッド最小化のため処理を割り当てず，不要な並列処理オーバーヘッドを削減している．

2.5 プロセッサグループへのマクロタスク割り当て

上述のように各マクロタスクグラフに合わせて生成したプロセッサグループがそのマクロタスクグラフを処理する単位となり，当該マクロタスクグラフ上の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コンパイル時に決定される．マクロタスクグラフが条件分岐等の実行時不確定性を含む場合は，実行時にタスクをプロセッサグループに割り当てる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むダイナミックスケジューリング方式がとられる．

2.6 粗粒度タスク間キャッシュ最適化

OSCAR コンパイラでは，データローカライゼーション手法を用いた粗粒度タスク間キャッシュ最適化[17] によりキャッシュの利用効率を高め，マルチグレイン並列処理の性能を向上させている．データローカライゼーション手法では，まずキャッシュサイズを超えるような大きなデータを持つ複数のループに対してループ整合分割[18] を適用し，キャッシュメモリに収まるようにそれぞれのループのイタレーション空間とデータを分割する．この際に複数ループ間のデータ依存を解析し，分割後の小ループ間でのデータ授受ができる限りキャッシュを介して行わるよう分割するイタレーション数，データサイズを調節する．この様に分割されたループのうち，同一データにアクセスする分割ループ集合をデータローカライザブルグループ（DLG）と呼ぶ．1 つの DLG 内でアクセスされる総データ量はキャッシュサイズ以下となるため，DLG 内のマクロタスク (MT) を同一のプロセッサで連続的に処理することによりキャッシュ上のデータの再利用性を高めることが可能となる．このため，粗粒度タスクスケジューリングでは粗粒度タスク間の並列性と DLG によるデータ局所性の両方を考慮しながらスケジューリングを行う．図 4(a) のマクロタスクグラフ (MTG) において，MT2，3，7 はキャッシュサイズを上回るデータを共有する並列化ループであるとし，図 4(b) のようにループ整合分割により MT i は MT i_A，MT i_B，MT i_C，MT i_D の 4 つの小ループにそれぞれ分割される．MT2_A，MT3_A，

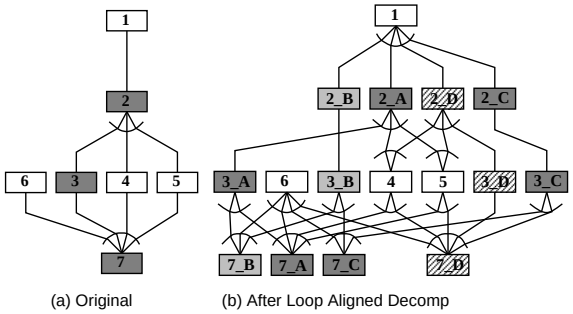


図 4 粗粒度タスク間キャッシュ最適化

```
!$OMP PARALLEL SECTIONS
!$OMP SECTION
    CALL MAIN_PE0
!$OMP SECTION
    CALL MAIN_PE1
!$OMP END PARALLEL SECTIONS
```

図 5 PARALLEL SECTIONS によるスレッド生成

MT7_A は DLG_A を構成し，同様に DLG_B，DLG_C，DLG_D が定義される．これら DLG 内でアクセスされる総データはキャッシュサイズ以下であり，DLG を同一のプロセッサで処理するように実行することでキャッシュミスの削減が望める．

さらにコンパイル時にキャッシュ上でのデータレイアウトを推定し，DLG 内ループによってアクセスされる配列間でのキャッシュラインコンフリクトの発生可能性を検出する．ラインコンフリクトが生じる場合，ダイレクトマップやセットアソシアティブキャッシュの way 数，way サイズを考慮して配列間パディングを行うことによりコンフリクトミスを削減する[19]．

従来のコンパイラによるキャッシュ最適化手法は主に単一ループや融合されたループを対象としているのに対して，本手法ではオリジナルプログラム中で離れた位置にある異なる複数のループ間でのキャッシュ最適化も可能となる．

2.7 OpenMP バックエンドによる並列化コード生成

OSCAR コンパイラは，実行するターゲットマシンに応じた複数のバックエンドを持つ．このうち，本論文では OpenMP API によって並列化されたプログラムを出力する OpenMP バックエンドを用い，生成された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，実行する．OpenMP バックエンドでは PARALLEL SECTIONS ディレクティブを用いることでプログラムの開始時にプロセッサ台数分の並列スレッドを生成する (図 5)．スタティックスケジューリングが適用された MTG に関しては，各スレッドが実行する MT はコンパイル時に決定され，それぞれのスレッド用のコード (図 5 中の MAIN_PE0 及び MAIN_PE1) 中には必要な MT のみが生成される．ダイナミックスケジューリングが適用された MTG では，各スレッド用コードは実行する可能性のある MT 全てを保持し，コンパイラが生成したスケジューラコードにより実行時に処理をする MT が決定され，その

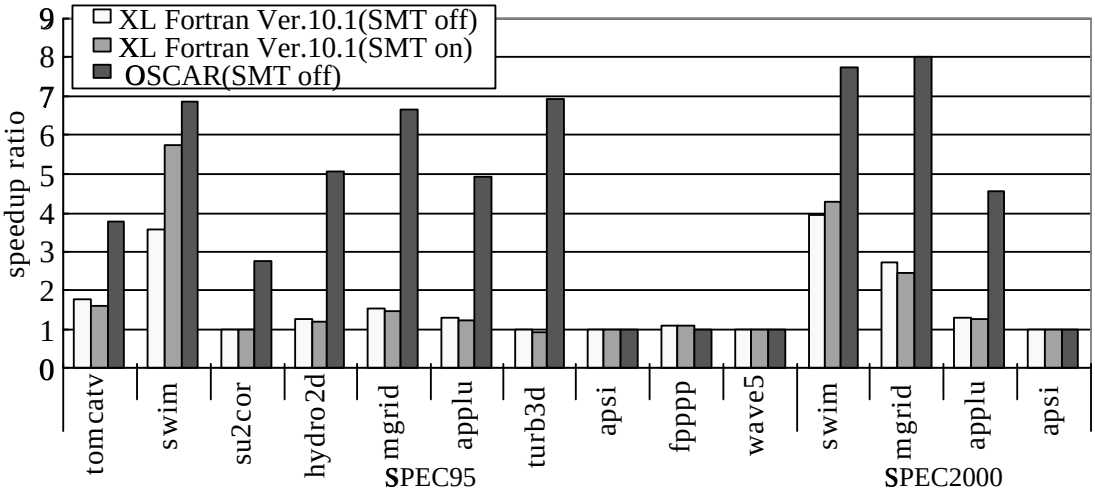


図 6 p5 550Q 上での速度向上率

MT のコードへ制御を移す．本バックエンドが必要とする OpenMP の機能としては上記の PARALLEL SECTIONS 以外にメモリアクセス順を保証する FLUSH，排他制御を行うための CRITICAL のみであり，これら 3 つからなる OpenMP API のサブセットがサポートされている環境ならば OSCAR コンパイラによるマルチグレイン並列処理が実現できる．

3. 性能評価

本章では SPEC CFP95 ベンチマークを用いた OSCAR マルチグレイン自動並列化コンパイラの性能評価について述べる．

3.1 p5 550Q 上の性能

図 6 に IBM XL Fortran コンパイラ version 10.1 と OSCAR コンパイラの，IBM p5 550Q 上での評価結果を示す．p5 550Q は Power5+ 2 コア集積マルチコアプロセッサを 4 つ搭載した 8 プロセッサ SMP サーバであり，1 プロセッサ当り 2 スレッドの Simultaneous Multi-Threading (SMT) が可能である．評価に用いたベンチマークは SPEC CFP95 全 10 本と SPEC CFP2000 171.swim，172.mgrid，173.applu，301.apsi である．図中，横軸が実行したベンチマークを表し，縦軸が IBM XL コンパイラ version 10.1 の逐次処理に対する速度向上率を表す．各ベンチマークにおいて左側のバーが SMT を用いない場合の XL コンパイラ version 10.1 の自動並列化性能であり，8 プロセッサまで用いたうちの最高性能を表す．中央のバーが SMT を用いた場合の XL コンパイラの最高性能，右側のバーが OSCAR コンパイラによる自動並列化の最高性能を表す．

OSCAR コンパイラによる並列処理の速度向上率は逐次処理に比べて SPEC CFP95 tomcatv で 3.77 倍，swim で 6.86 倍，su2cor で 2.72 倍，hydro2d で 5.07 倍，mgrid で 6.67 倍，applu で 4.93 倍，turb3d で 6.95 倍であり，SPEC CFP2000 swim で逐次処理に比べ 7.74 倍，CFP2000 mgrid で 8.01 倍，CFP2000 applu で 4.58 倍であった．XL コンパイラの並列処理に対する OSCAR コンパイラの速度向上率は，SPEC CFP95 tomcatv において SMT を用いない XL コンパイラの並列処理に比べ 2.17 倍，SMT を用いた場合に比べ 2.41 倍であった．また swim では SMT を用いない場合に比べ 1.92 倍，SMT を用いた場合に比べ 1.19 倍であり，su2cor ではそれぞれ 2.72 倍と 2.72 倍，hydro2d では 4.10 倍と 4.29 倍，mgrid では 4.34 倍と 4.58 倍，applu では 3.79 倍と 4.03 倍，turb3d では 6.95 倍と 6.95 倍の速度向上が得られた．SPEC CFP2000 swim では SMT を用いない場合に比べ 1.96 倍，SMT を用いた場合に比べ 1.80 倍であり，CFP2000 mgrid ではそれぞれ 2.96 倍と 3.29 倍，CFP2000 applu では 3.52 倍と 3.70 倍の速度向上が得られた．

tomcatv, swim, hydro2d, mgrid はループ並列性の高いベンチマークであるが，図 6 のように OSCAR コンパイラの並列化性能は全ベンチマークの平均で，SMT を用いない XL コンパイラ version 10.1 に比べて 2.74 倍，SMT を用いた場合に比べて 2.78 倍と XL コンパイラを大きく上回っている．これは OSCAR コンパイラでは粗粒度タスク並列化が実現されていること，グローバルなキャッシュ最適化が行われていること，並列処理階層自動決定手法によりプログラム中の各部分に対して並列性に見合った処理プロセッサ数で実行し，過剰なプロセッサによる無駄な並列処理オーバーヘッドの発生を回避していること，また各並列スレッドをコンパイラが生成したスケジューリングコードが制御しているため，OS あるいはランタイムルーチンによる制御に比べスケジューリングオーバヘッドの抑制が可能となっているためである．一方，apsi，fpppp，wave5 では並列化可能ループがないあるいはループサイズが小さいため市販サーバ上での並列化が困難であり，OSCAR コンパイラ，XL コンパイラともに並列処理による速度向上は得られなかった．ただしこれらのアプリケーションもマルチコアプロセッサのようにプロセッサ間データ転送オーバーヘッドが小さくより細かい粒度の並列性利用が可能なシステム上では OSCAR コンパイラによる性能向上が望める[20]．

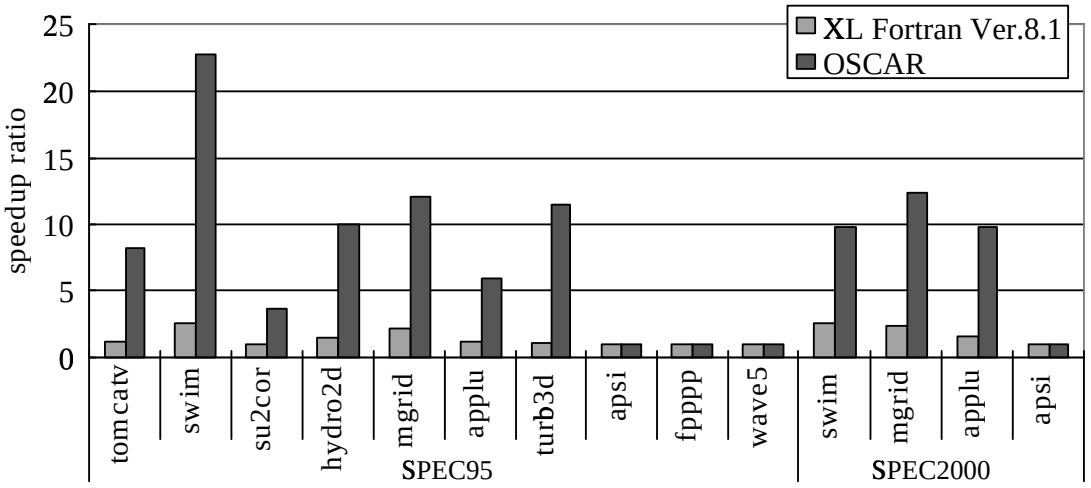


図 7 pSeries690 上での速度向上率

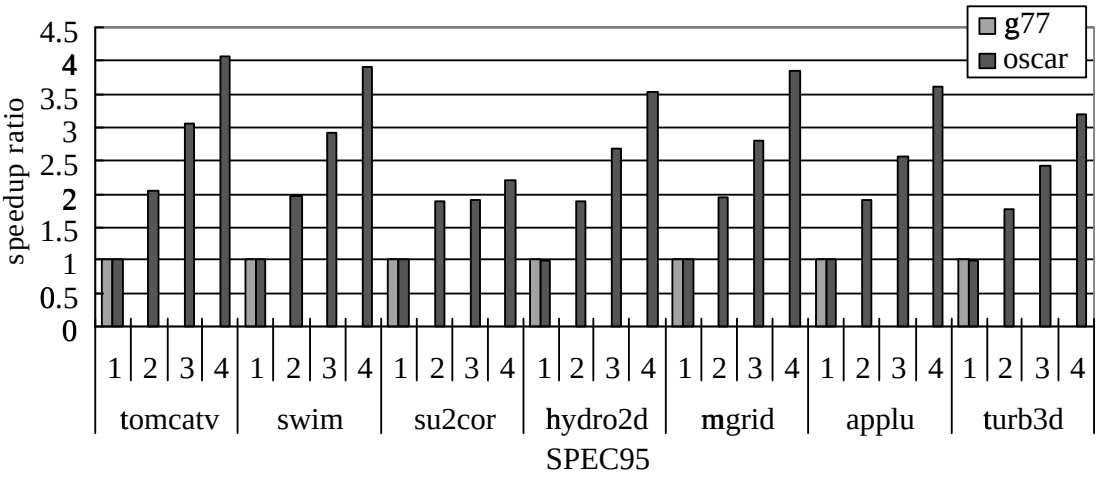


図 8 MPCore 上での速度向上率

3.2 pSeries690 上の性能

図 7 に 1.1 GHz の Power4 24 way ハイエンドサーバ IBM pSeries690 上での，IBM XL Fortran コンパイラ version 8.1 と OSCAR コンパイラの評価結果を示す．左側のバーが XL コンパイラの 24 プロセッサまで用いたうちの最高性能，右側のバーが OSCAR コンパイラの 24 プロセッサまでの最高性能を表す．

OSCAR コンパイラによる並列処理の速度向上率は tomcatv では逐次処理に比べ 8.15 倍，XL コンパイラの最速並列処理に比べ 6.76 倍，CFP95 swim では逐字に対し 22.8 倍，XL コンパイラに対し 9.13 倍，su2cor ではそれぞれ 3.66 倍と 3.66 倍，hydro2d では 10.0 倍と 6.94 倍，mgrid では 12.1 倍と 5.65 倍，applu では 5.88 倍と 4.92 倍，turb3d では 11.5 倍と 10.9 倍の速度向上が得られた．また SPEC CFP2000 swim でそれぞれ 9.82 倍と 3.72 倍，CFP2000 mgrid で 12.3 倍と 5.35 倍，CFP2000 applu で 9.87 倍と 6.55 倍の速度向上が得られた．

apsi，fpppp，wave5 では p5 550Q と同様な理由で並列処理による速度向上は得られなかった．SPEC CFP95 10 本と CFP2000 4 本の平均では，OSCAR コンパイラは XL コンパイラ version 8.1 に比べて 4.82 倍の性能向上が得られた．

3.3 MPCore 上の性能

次に NEC/ARM の 4 プロセッサ集積組込み用マルチコアである MPCore チップ上での，OSCAR コンパイラの自動並列化性能について述べる．今回の評価環境では OpenMP の内，OSCAR コンパイラが必要とするスレッド生成 SECTION，排他制御 CRITICAL，FLUSH のみの機能を実現する並列化ライブラリを用意し，評価を行った．今回使用した MPCore テストボードの主メモリアクセスに制限があるため，データサイズを縮小した SPEC CFP95 tomcatv，swim，su2cor，hydro2d，mgrid，applu，turb3d を評価に用いた．表 1 に本評価に用いた MPCore テストボードの CPU 周波数と各種メモリサイズを示す．

g77 コンパイラによる逐次処理に対する，OSCAR コンパイラの 4 プロセッサでの並列処理性能は図 8 に示すように tomcatv で 4.08 倍，swim で 3.90 倍，su2cor で 2.21 倍，hydro2d で 3.53 倍，mgrid で 3.85 倍，applu で 3.62 倍，turb3d で 3.20 倍の速度向上率となり，MPCore 上での OSCAR コンパイラによる自動マルチグレ

表 1 MPCore テスチップのパラメータ

CPU 周波数	200[MHz]
L1 I キャッシュ	32[KB]
L1 d キャッシュ	32[KB]
L2 共有キャッシュ	1[MB]
メインメモリ	128[MB]

イン並列化の有効性が確認できた．

4. ま と め

本論文では OSCAR マルチグレイン自動並列化コンパイラの Power5+ ベース最新 SMP サーバ p5 550Q および組込み向け SMP マルチコア MPCore 上での性能について述べた．OSCAR コンパイラは各粒度の並列性に応じた適切な計算資源割当て及びデータレイアウト変換パディング技術を用いたデータローカライゼーション技術，低オーバーヘッドコード生成法を用いた並列処理を実現している．

SPEC CFP95 および CFP2000 を用いた評価では，OSCAR コンパイラによる自動並列化は IBM p5 550Q Power5+ 8 プロセッササーバ上で IBM XL Fortran コンパイラ version 10.1 の自動並列化性能に比べ平均 2.74 倍，IBM pSeries690 Power4 24 プロセッササーバ上で IBM XL Fortran コンパイラ version 8.1 の自動並列化性能に比べ平均 4.82 倍，NEC/ARM MPCore ARMv6 4 プロセッサ組込み用マルチコア上で世界で初めて自動並列化に成功し，g77 の逐次処理に比べ tomcatv で 4.08 倍，swim で 3.90 倍，su2cor で 2.21 倍，hydro2d で 3.53 倍，mgrid で 3.85 倍，applu で 3.62 倍，turb3d で 3.20 倍といった大きなスピードアップが得られることが確かめられた．これにより OSCAR コンパイラで用いられている並列化技術の SMP サーバ及び組込み向けマルチコア上での十分な実用性が確かめられた．

5. 謝　　辞

本研究の一部は NEDO “リアルタイム情報家電用マルチコア技術”，NEDO “先進ヘテロジニアスマルチプロセッサ研究開発” 及び STARC（半導体理工学研究センター）“並列化コンパイラ協調型チップマルチプロセッサ技術” の支援により行われた．

参 考 文 献

1) Pham, D. et al.: The Design and Implementation of a First-Generation CELL Processor, *In Proceeding of the IEEE International Solid-State Circuits Conference* (2005).
2) Cornish, J.: Balanced Energy Optimization, *International Symposium on Low Power Electronics and Design* (2004).
3) Suga, A. and Matsunami, K.: Introducing the FR 500 embedded microprocessor, Vol. 20, pp. 21–27 (2000).
4) Intel: http://www.intel.com/multi-core/.
5) Keltcher, C. N., McGrath, K., Ahmed, A. and Conway, P.: The AMD Opteron processor for multiprocessor servers, *IEEE Micro*, Vol. 23, pp. 66–76 (2003).
6) Clabes, J. et al.: Design and implementation of the POWER5 microprocessor, *IEEE ISSCC*, pp. 55–57 (2004).
7) Wolfe, M.: High Performance Compilers for Parallel Computing, *Addison-Wesley Publishing Company* (1996).
8) Eigenmann, R., Hoeflinger, J. and Padua, D.: On the Automatic Parallelization of the Perfect Benchmarks, *IEEE Trans. on parallel and distributed systems*, Vol. 9, No. 1 (1998).
9) Hall, M. W., Anderson, J. M., Amarasinghe, S. P., Murphy, B. R., Liao, S., Bugnion, E. and Lam, M. S.: Maximizing Multiprocessor Performance with the SUIF Compiler, *IEEE Computer* (1996).
10) Gonzalez, M., Martorell, X., Oliver, J., Ayguade, E. and Labarta, J.: Code Generation and Runtime Support for Multi-level Parallelism Exploitation, *Proc. of the 8th International Workshop on Compilers for Parallel Computing* (2000).
11) Saito, H., Stavakos, N. and Polychronopoulos, C.: Multithreading Runtime Support for Loop and Functional Parallelism, *Proc. of the International Symposium on High Performance Computing* (1999).
12) 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).
13) Kasahara, H. et al.: A Multi-grain Parallelizing Compilation Scheme on OSCAR, *Proc. 4th Workshop on Language and Compilers for Parallel Computing* (1991).
14) 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).
15) 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処理制御手法, 情報処理学会論文誌, Vol. 44, No. 4 (2003).
16) 白子準, 長澤耕平, 石坂一久, 小幡元樹, 笠原博徳: 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, 情報処理学会論文誌, Vol. 45, No. 5 (2004).
17) 石坂, 中野, 八木, 小幡, 笠原: 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 (2002).
18) 吉田, 前田, 尾形, 笠原: 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手法, 情報処理学会論文誌, Vol. 35, No. 9, pp. 1848–1994 (1994).
19) 石坂一久, 小幡元樹, 笠原博徳: 配列間パディングを用いた粗粒度タスク間キャッシュ最適化, 情報処理学会論文誌, Vol. 45, No. 4 (2004).
20) 近細粒度並列処理用シングルチップマルチプロセッサにおけるプロセッサコアの評価: 木村 啓二 and 加藤 孝幸 and 笠原 博徳, 情報処理学会論文誌, Vol. 42, No. 4 (2001).