OSCAR コンパイラにおける制約付き C プログラムの自動並列化

間瀬正啓[†] 馬場大介[†] 長山晴美[†]
田野裕秋[†] 益浦健[†] 深津幸二[†]
宮本孝道[†] 白子準[†] 中野啓史[†]
木村啓二[†] 笠原博徳[†]

マルチプロセッサ，マルチコアアーキテクチャの普及に伴い，ハイパフォーマンスコンピューティング分野から組み込み分野に至る様々な分野で，その特性を引き出し高実効性能・低消費電力を実現する自動並列化コンパイラの重要性が高まっている．本稿ではプログラム全域の並列性及びデータローカリティの有効利用が可能なマルチグレイン並列処理を実現する，OSCAR コンパイラの C 言語対応について述べる．OSCAR コンパイラにおける C 言語対応を迅速に行うために制約付き C 言語を定めた．MPEG2 エンコード，MP3 エンコード，AAC エンコードの各メディアアプリケーション，組み込み向けベンチマーク MiBench より susan (smoothing)，SPEC2000 より art について C 言語対応 OSCAR コンパイラによる自動並列化の初期性能評価を行い，8 プロセッササーバである IBM p5 550 上で IBM XL C コンパイラ version 8.0 の逐次処理と比較して susan (smoothing) で最大 7.49 倍，4 プロセッサワークステーションである Sun Ultra80 上で Sun Studio 9 C コンパイラの逐次処理と比較して susan (smoothing) で最大 3.75 倍の速度向上が得られた．

Automatic Parallelization of Restricted C Programs in OSCAR Compiler

Masayoshi Mase,[†] Daisuke Baba,[†] Harumi Nagayama,[†]
Hiroaki Tano,[†] Takeshi Masuura,[†] Koji Fukatsu,[†]
Takamichi Miyamoto,[†] Jun Shirako,[†] Hirofumi Nakano,[†]
Keiji Kimura[†] and Hironori Kasahara [†]

Along with the popularization of multiprocessors and multicore architectures, automatic parallelizing compiler, which can realize high effective performance and low power comsumption, becomes more and more important in various areas from high performance computing to embedded computing. OSCAR compiler realizes multigrain automatic parallelization, which can exploit parallelism and data locality from the whole of the program. This paper describes C language support in OSCAR compiler. For rapid support of C language, restricted C language is proposed. In the preliminary performance evaluation of automatic parallelization using following media applications as MPEG2 encode, MP3 encode, and AAC encode, Susan (smoothing) derived from MiBench, and Art from SPEC2000, OSCAR compiler achieved 7.49 times speed up in maximum for susan (smoothing) against sequential execution on IBM p5 550 server having 8 processors, and 3.75 times speed up in maximum for susan (smoothing) too against sequential execution on Sun Ultra80 workstation having 4 processors.

1. はじめに

マルチプロセッサ・マルチコアアーキテクチャはワークステーションやハイエンドサーバといったハイパフォーマンスコンピューティング分野から携帯電話，PDA，ゲーム等の組み込み分野まで幅広く利用されている．このようなマルチプロセッサシステムにおいて高い実効性能を実現するためには，プログラム全域からの適切な粒度の並列性抽出，プロセッサ近傍の高速キャッシュメモリあるいはローカルメモリの最適利用が必須であり，これを実現するための自動並列化コンパイラの研究・開発が行われている[1)~3)]．これらの研究・開発の大部分はプログラム中のループ部分の並列化を対象としたものであり，現在までに様々なループ並列性解析手法やリストラクチャリング手法が開発されている．ループ並列化手法は大きな進歩を遂げたが，現在では既に成熟期に至り今後の大幅な性能向上は見込めないと考えられている．そのため，マルチプロセッサシステムの更なる実効性能向上のためには，従来のループ並列性に加え，ループ間やサブルーチン間といった粗粒度タスク並列性や，基本ブロック内での命令・ステートメント間の近細粒度並列性などの複数レベルの並列性を利用することが必須である．早稲田大学で開発されている OSCAR コンパイラ[4)~7)] では，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．OSCAR コンパイラはこれまで科学技術計算分野で広く用いられている FORTRAN77 言語を対象として開発を行ってきたが，より広範囲のアプリケーション，特に組み込み分野における自動並列化技術の適用に対する要求から C 言語への対応を進めている．一般的に C 言語はポインタの明示的な使用等の言語仕様から最適

† 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

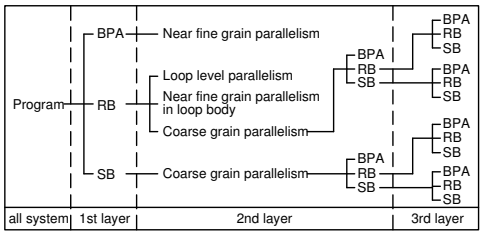


図 1 階層的マクロタスク定義

化が困難な言語とされており[8]，OSCAR コンパイラにおける C 言語対応を迅速に行うために制約付き C 言語を定め，初期段階の評価として入力ソースプログラムに一定の制約条件を設けることにより短期間のうちに自動並列化を実現した．

本稿では OSCAR コンパイラの C 言語対応，制約付き C 言語，及び初期段階の性能評価として制約付き C プログラムの SMP マシン上での自動並列化について述べる．

本稿の構成は以下のようになる．まず 2 章で OSCAR コンパイラの基盤技術であるマルチグレイン並列処理について述べ，第 3 章で OSCAR コンパイラの構成について述べる．そして第 4 章で自動並列化のための制約付き C 言語について述べ，第 5 章で制約付き C プログラムの SMP マシン上での性能評価について述べる．最後に第 6 章で本稿のまとめを述べる．

2. マルチグレイン並列処理

本章では，OSCAR コンパイラで実現されてい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粗粒度タスク並列性，ループ並列性，近細粒度並列性を組み合わせ，プログラム全域から並列性を抽出する技術である．

本稿で評価に用いたような商用 SMP サーバでは，低レイテンシのプロセッサ間データ通信機構が必要な近細粒度並列処理[9] は同期及びデータ転送のオーバーヘッドが大きいため，本稿では粗粒度タスク並列性とループ並列性を用いたマルチグレイン並列処理を行う．

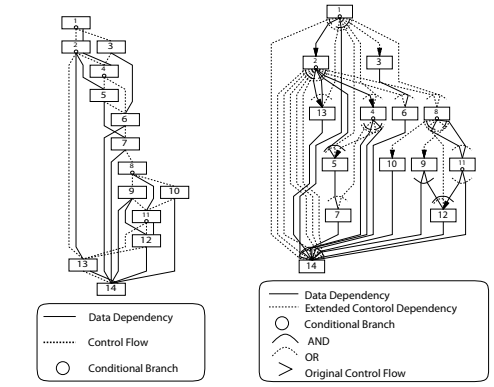
2.1 粗粒度タスク生成

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[6]，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[6]，サブルーチンブロック SB[6] の 3 種類の粗粒度タスク (マクロタスク MT[6]) 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 は図 1 に示すようにその内部をさらにマクロタスクに分割し階層的なマクロタスク構造を生成する．

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，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[4),6)] 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[4),6)] を適用し，階層的なマクロタスクグラフ MTG[4),6)] を生成する．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は粗粒度タスク並列性を表す．マクロフローグラフ及びマクロタスクグラフの例を図 2 に示す．



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

図 2 マクロフローグラフとマクロタスクグラフ

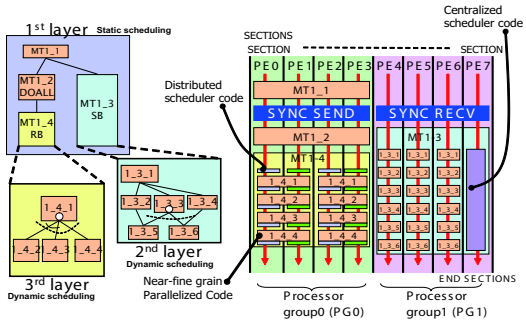


図 3 階層的コード生成イメージ

2.3 プロセッサグループへのマクロタスク割り当て

コンパイラはマクロタスクを各プロセッサエレメント PE[10] あるいは PE を複数集めたプロセッサグループ PG[10] 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等の実行時不確定性を含む場合は実行時にスケジューリングを行なうダイナミックスケジューリングルーチンをコンパイラが自動生成し，実行時にマクロタスクを PE あるいは PG に割り当てる．図 3 に示すように各マクロタスクは階層的にスタティックスケジューリングあるいはダイナミックスケジューリングされる．生成されたスタティックスケジューリングコード及び実行時スケジューラはユーザコードであり，OS のシステムコールによるスケジューラに比べ極めて低オーバーヘッドなスケジューリングが可能である．

2.4 データローカライゼーション

プロセッサとメモリの速度差の拡大によりキャッシュメモリやローカルメモリを有効利用することがマルチプロセッサシステムの性能向上にとって重要となっている．OSCAR コンパイラでは並列性とデータローカリティの両方を考慮したデータローカライゼーション手法[11] により複数粗粒度タスク間でキャッシュあるいはローカルメ

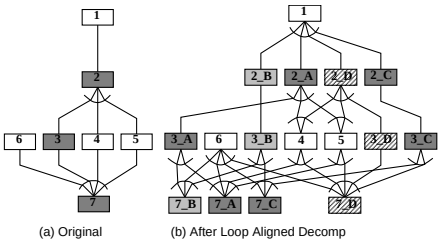


図 4 データローカライゼーションにおけるループ整合分割

モリ上のデータを効果的に用いる．

データローカライゼーション手法では，まず複数ループ間のデータ依存を解析し，データ依存する分割後の小ループ間におけるデータ授受がキャッシュあるいはローカルメモリを介して行われるようにそれらのループを整合して分割するループ整合分割[12] を行う．分割されたループのうち同一データにアクセスする複数のマクロタスクは，データローカライザブルグループ（DLG）と呼ぶタスク集合にグループ化される．図 4 にループ整合分割を適用したマクロタスクグラフを示す．図中 (b) の同じ網掛けで塗られたマクロタスクが DLG に属するマクロタスクである．

整合分割後の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を考慮しながら，同一 DLG に属するマクロタスクが可能な限り同一プロセッサ上で連続的に実行されるようにスケジューリングを行う．このようにループ分割と DLG 内タスクの連続実行を組み合わせることにより，複数のループに渡り再利用することを可能とすることでメインメモリアクセスを削減し，タスク間のデータ授受をキャッシュあるいはローカルメモリを用いて高速に行うことが可能となる．

3. OSCAR コンパイラの構成

本章ではマルチグレイン自動並列化を実現する OSCAR コンパイラの構成[7] について述べる．C 言語対応において機能拡張を行った点を中心に簡単に説明する．図 5 に示すように OSCAR コンパイラはフロントエンド，ミドルパス，バックエンドの 3 つのフェーズから構成されている．また，フェーズ間の入出力や各種最適化は中間表現に対して行われる．

3.1 フロントエンド (FE)

フロントエンドはソースプログラムの字句解析及び構文解析を行い，逐次のプログラムをコンパイラの中間表現に変換する．C 言語対応にあたり CoSy コンパイラ開発システム[13] を用いて C フロントエンドを開発した．本フロントエンドは C99[14] に対応している．

3.2 ミドルパス (MP)

ミドルパスではフロントエンドの生成した中間表現を入力し，制御フロー解析，データ依存解析等の各種解析をプログラム全域に渡って行う．これらのグローバルな解析結果に基づいてマルチグレイン並列化，データローカライゼーション，低消費電力制御[15] 等の最適化を行い，並列化された中間表現を出力する．C 言語対応にあたり拡張された中間表現を扱うための機能拡張を行った．

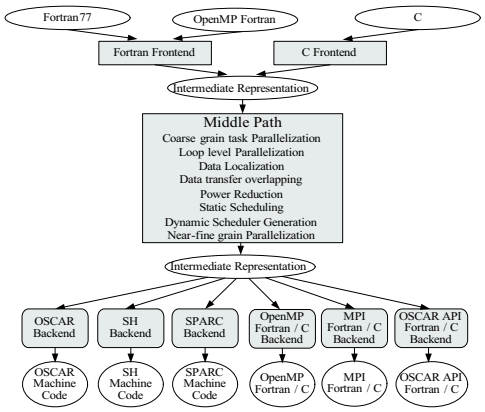


図 5 OSCAR コンパイラの構成

3.3 バックエンド (BE)

バックエンドは並列化された中間表現を入力とし，ターゲットマシン用のマシンコードまたは並列処理用に拡張された FORTRAN や C 言語のソースコードを出力する．対象に応じて独立したバックエンドを持ち，また並列処理用 API を利用することにより，多様なアーキテクチャ，プラットフォームに対応することを可能としている．本稿の評価では新たに開発した OpenMP C バックエンドを用いた．

3.4 中間表現

OSCAR コンパイラの中間表現においてプログラムは関数，変数，定数等のシンボルテーブルおよび四つ組により記述される．C 言語対応において FORTRAN77 には存在しなかった unsigned 型，char 型，ポインタ・構造体等の各種データ型，ポインタ・構造体参照やビットシフト演算等の各種オペレータ，グローバル変数，標準ライブラリ関数およびユーザプログラム中で定義されない関数等への対応を行った．本中間表現は C99[14] に対応している．

4. 自動並列化のための制約付き C 言語

本章では OSCAR コンパイラの C 言語対応について述べる．C プログラムの自動並列化を行うにあたり，まず FORTRAN77 のレベルまで言語仕様を制限した制約付き C 言語を規定し，これを並列化することから C 言語対応を開始した．このような方針により迅速な C 言語対応が可能となった．この制約付き C 言語では C 言語の記述に対して制限，あるいはディレクティブによるヒント情報の指定を行うことで並列性の抽出を容易にし，OSCAR コンパイラによる自動並列化を実現する．本制約を満たすようにプログラムを記述することで，OSCAR コンパイラにおいてプログラムの持つ並列性を最大限利用することが可能となる．また，ディレクティブを無視することで，通常の C プログラムとして処理することも可能である．現在の制約付き C 言語の特徴を以下に示す．

分割コンパイル

OSCAR コンパイラはプログラム全域からの並列性，データローカリティの抽出を行うためコンパイル時に全てのユーザプログラムを一度にコンパイルする

```
#pragma mc ARRAY a,@(20,10), b,@(20,10)
#pragma mc ARRAY c,@(20,10)
#pragma mc SCALAR d
int func(int a[][10], int b[][10], int c[][10], int *d){
        a[2][3] = ···;
        ··· = ··· b[3][2];
        *d = ···
        ··· = ··· *d;
}
```

図 6 関数のポインタ引数ディレクティブ

必要がある．ライブラリ関数の使用については，数学ライブラリ等の内部状態を持たない標準ライブラリ関数を除き，ライブラリ関数を含む部分の並列化は行わない．

関数の再帰呼び出し

関数の再帰呼出しは行わない．

ポインタ・構造体

ポインタ・構造体は原則的に使用しない．ヒープについても可能な限り単純な多次元配列を用いて代替する．ただし，後述する関数のポインタ引数ディレクティブで指定されたポインタについては例外とする．現在の OSCAR コンパイラではポインタ・構造体アクセスは，全てのメモリ領域に対してアクセスする可能性があるものとして扱う．

関数のポインタ引数ディレクティブ

配列を実引数として関数呼出しを行う場合 C の言語仕様では仮引数はポインタとなり，実引数の配列としての情報が失われてしまう．さらに，FORTRAN77 のように実引数と仮引数を静的にエイリアスすることができなくなってしまう．そこで，これらの情報を補うディレクティブを関数の直前に記述する．ポインタ引数ディレクティブの例を図 6 に示す．図中の ARRAY ディレクティブの変数名に続く部分は，関数 “func” においてポインタ引数を多次元配列とみなした場合の各次元の宣言サイズである．またポインタ引数ディレクティブ指定を行った引数については，ポインタ引数への値の再代入は行わない，C99[14] の restrict 修飾子と同様に複数のポインタ引数を用いた参照先が重ならない，という制限があるものとする．ポインタ引数ディレクティブにより，コンパイラではポインタ仮引数を FORTRAN77 における参照渡しによる関数呼出しの仮引数と同様に扱うことができる．

5. 性 能 評 価

本章では制約付き C プログラムを用いた C 言語対応 OSCAR コンパイラの初期性能について述べる．

5.1 評 価 条 件

商用 SMP マシンである IBM p5 550 および Sun Ultra80 上で OSCAR コンパイラと各マシン用のネイティブコンパイラの性能評価を行った．ネイティブコンパイラの評価時には，制約付き C 言語のディレクティブ指定は無視され一般の逐次 C プログラムとして自動並列化が適用される．OSCAR コンパイラの評価については，ミドルパスにおける C 言語対応が開発途中のため，制約付き C 言語の関数のポインタ引数ディレクティブ，およびデータローカライゼーションのためのディレクティブによるヒント情報を指定し，自動並列化を適用した．OSCAR コンパイラの自動並列化コードは OpenMP C バックエンドを用いて並列化された OpenMP C プログラムとして出力し，ネイティブコンパイラでコンパイルし実行した．本評価においては並列処理性能を評価するために，実行環境に著しく依存する I/O 処理の時間を除外し，演算処理部分のみを評価の対象としている．

5.2 対象アプリケーション

MP3 エンコード, MPEG2 エンコード, AAC エンコードの各メディアアプリケーション，組み込み向けベンチマーク MiBench より susan (smoothing)，SPEC2000 より art を用いて評価を行った．

MPEG2 エンコードは MediaBench[16] に収録されている MPEG2 エンコードプログラムである “mpeg2encode” を制約付き C で参照実装したプログラムを用いた．入力画像は SIF サイズの NHK の標準動画像[17] より “瓶と果物” を用い，エンコードを行った．エンコードオプションは MediaBench のデフォルトパラメータと同一とした．

MP3 エンコードは UZURA3: MPEG1/LayerIII Encoder in FORTRAN90[18] を制約付き C で参照実装したプログラムを用いた．入力データはサンプリングレート 44.1kHz のステレオ PCM データ，出力データのビットレートは 128kbps とし，その他のエンコードオプションは参照した UZURA のデフォルトパラメータと同一とした．

AAC エンコードは株式会社 ルネサス テクノロジ提供のアプリケーションであり，製品ミドルウェア仕様を並列性抽出が可能となるように制約付き C 言語で参照実装したものとなっている．入力データはサンプリングレート 44.1kHz のステレオ PCM データ，出力データのビットレートは 96kbps とした．

susan は 組み込み向けベンチマーク MiBench[19] に収録されている “susan” を制約付き C 言語仕様を満たすように修正したプログラムを用いた．susan は画像認識アプリケーションであり，smoothing, edges, corners の 3 種類のモードがあるが，本稿の評価では特に大きな並列性のある smoothing について評価を行った．

art は SPEC2000 に収録されている，“179.art” を制約付き C 言語仕様を満たすように修正したプログラムを用いた．art はニューラルネットワークを用いた画像認識アプリケーションである．データサイズは ref を用いた．

5.3 IBM p5 550 上での評価

図 7 に IBM XL C コンパイラ version 8.0 と OSCAR コンパイラを IBM p5 550 上で評価した結果を示す．図中，横軸が評価を行ったアプリケーションおよび使用したプロセッサ数を示し，縦軸が IBM XL C コンパイラ version 8.0 の逐次処理に対する速度向上率を示す．それぞれ左側のバーが XL C コンパイラの自動並列化による速度向上率，右側のバーが OSCAR コンパイラの自動並列化による速度向上率を示す．

IBM p5 550 は Power5+ 2 コアを集積したマルチコア

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 チップ (2 プロセッサ) あたり 1.9 MB の L2 キャッシュ，36MB の L3 キャッシュを搭載している．1 プロセッサあたり 2 スレッドの Simultaneous Multi-Threading (SMT) が可能であるが，本評価では SMT は用いないものとした．使用したネイティブコンパイラは IBM XL C コンパイラ version 8.0 でありコンパイルオプションは，OSCAR コンパイラが生成した OpenMP C ソースのコンパイル時は “-O5 -qsmp=noauto”，ネイティブコンパイラによる自動並列化では”-O5 -qsmp=auto” を用いた．

OSCAR コンパイラによる並列処理の速度向上率は 8 プロセッサ用いた際に IBM XL C コンパイラ version 8.0 の逐次処理と比較して MPEG2 エンコードで 5.19 倍，MP3 エンコードで 3.69 倍，AAC エンコードで 7.41 倍，susan (smoothing) で 7.49 倍，art で 3.76 倍であった．一方，XL C コンパイラ version 8.0 の自動並列化ではいずれのアプリケーションも速度向上は得られなかった．

AAC エンコード，susan (smoothing) について OSCAR コンパイラでは 8 プロセッサにおいてそれぞれ 7.41 倍，7.49 倍と非常に大きな速度向上を得ることができた．この 2 つのアプリケーションでは演算処理の大部分を一つの doall ループが占めており，OSCAR コンパイラではこのループが doall ループと判定できたことが速度向上につながったと考えられる．

MPEG2 エンコードプログラムについても 8 プロセッサにおいて 5.19 倍と大きな速度向上が得られた．MPEG2 エンコードプログラムはマクロブロックレベルの並列性とデータローカリティ[20] を持っており，演算処理の大半を占める逐次ループの内部は，doall ループおよび逐次ループが連続したプログラム形状となっている．OSCAR コンパイラにおいてはこの並列性とデータローカリティを有効利用することができたと考えられる．

MP3 エンコードプログラムはフレームレベルの並列性，データローカリティを持ち，8 プロセッサにおいて 3.69 倍の速度向上が得られた．MPEG2 エンコードほどの性能向上を得られていないが，これはフレームのエンコード処理中に存在する収束ループにおいて，フレームによって演算時間のばらつきがあり，プロセッサ間に負荷の不均衡が生じたためと考えられる．

art においては 8 プロセッサにおいて 3.76 倍の速度向上が得られた．art は主要演算ループ内に大きな逐次処理部を含むため，プログラムの持つ並列性を最大限抽出できた結果と考えられる．

5.4 Sun Ultra80 上での評価

図 8 に Sun Studio 9 C コンパイラ と OSCAR コンパイラの Sun Ultra80 上での評価結果を示す．図中，横軸が実行したアプリケーションおよび使用したプロセッサ数を示し，縦軸が Sun Studio 9 C コンパイラの逐次処理に対する速度向上率を示す．それぞれ左側のバーが Sun Studio 9 C コンパイラの自動並列化による速度向上率，右側のバーが OSCAR コンパイラの自動並列化による速度向上率を示す．

Sun Ultra80 は 450MHz の UltraSPARCII プロセッサを 4 つ搭載した 4 プロセッサ SMP ワークステーションであり，1 プロセッサあたり各 16KB の L1 命令キャッシュおよび L1 データキャッシュ，4MB の L2 キャッシュを持つ．使用したネイティブコンパイラは Sun Studio 9 C コンパイラでありコンパイルオプションは，OSCAR コンパイラが生成した OpenMP C ソースのコンパイル時は “-fast -xopenmp”，ネイティブコンパイラによる自動並列化では”-fast -xautopar -xreduction” を用いた．

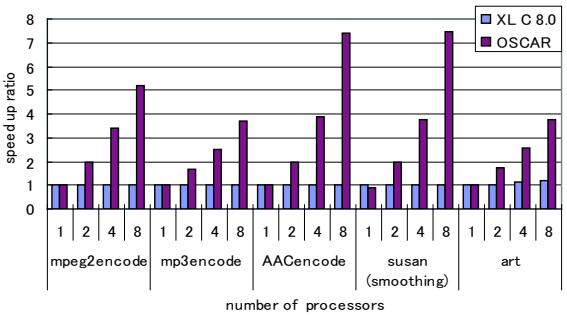


図 7 IBM p5 550 上の速度向上率

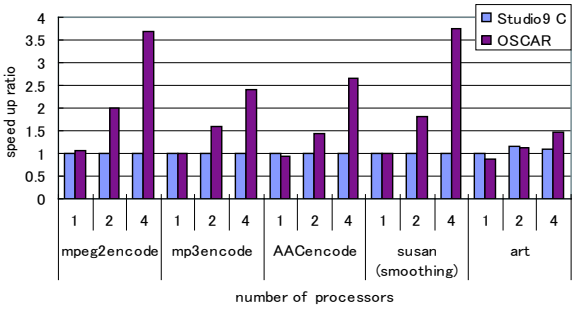


図 8 Sun Ultra80 上の速度向上率

OSCAR コンパイラによる並列処理の速度向上率は 4 プロセッサ用いた際に Sun Studio 9 C コンパイラの逐次処理と比較して MPEG2 エンコードで 3.68 倍，MP3 エンコードで 2.39 倍，AAC エンコードで 2.66 倍，susan (smoothing) で 3.75 倍，art で 1.47 倍であった．一方，Studio 9 C コンパイラの自動並列化では art で 2 プロセッサ時に 1.17 倍と若干の速度向上を得られたのを除き速度向上は得られなかった．

この結果を IBM p5 550 上での結果と比較すると AAC エンコード，art について性能向上率の鈍化が見られるが，IBM p5 550 が 2 プロセッサごとに 1.9 MB，10-way associative の L2 キャッシュおよび 36MB，12-way associative の L3 キャッシュを持つのに対し，Sun Ultra80 は 4MB，ダイレクトマップの L2 キャッシュを持つのみであり，キャッシュ性能の違いが原因の一つと考えられる．また 1 プロセッサで実行した際の性能を比較した際，art では OSCAR コンパイラにおいて Sun Studio 9 C コンパイラの 0.86 倍と逐次性能が低下しており，OSCAR コンパイラの OpenMP C バックエンドの出力コードに対するネイティブコンパイラの逐次最適化が，オリジナルソースほど適用されなかったことも要因と考えられる．

6. ま と め

本稿では OSCAR コンパイラにおける C 言語対応について述べた．C 言語への対応を迅速に行うため自動並列化のための制約付き C 言語を定め，実際に制約付き C 言語で記述されたプログラムに対して SMP マシン上で初期性能評価を行った．その結果，MPEG2 エンコード，MP3 エンコード，AAC エンコード, susan (smoothing), art の各アプリケーションについて，OSCAR コンパイラでは 8 プロセッサ SMP サーバである IBM p5 550 において XL C コンパイラ version 8.0 の逐次処理と比較して MPEG2 エンコードで 5.19 倍，MP3 エンコードで 3.69 倍，AAC エンコードで 7.41 倍，susan (smoothing) で 7.49 倍，art で 3.76 倍と，大きな速度向上が得られた．同様に 4 プロセッサ SMP ワークステーションである Sun Ultra80 においても Sun Studio 9 C コンパイラの逐次処理と比較して MPEG2 エンコードで 3.68 倍，MP3 エンコードで 2.39 倍，AAC エンコードで 2.62 倍，susan (smoothing) で 3.75 倍，art で 1.47 倍と，速度向上を得ることができた．これにより，制約付き C プログラムに対する OSCAR コンパイラの有効性が確かめられた．

今後は OSCAR コンパイラにおけるポインタ・構造体への対応を進め，並列化のための制約付き C 言語仕様の制約緩和を模索するとともに，組み込み向けマルチコアプロセッサ上での性能評価を行う予定である．

7. 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．

また，本稿で性能評価に用いた AAC エンコードプログラムをご提供いただきました株式会社ルネサステクノロジ様に感謝申し上げます．

参 考 文 献

1) M.Wolfe: High Performance Compilers for Parallel Computing, *Addison-Wesley Publishing Company* (1996).
2) 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).
3) 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).
4) 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).
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) 岡本, 小幡, 松崎, 笠原, 成田: マルチグレイン並列化 FOTRAN コンパイラ, 情報処理学会論文誌, Vol. 40, No. 12, pp. 4296–4308 (1999).
8) Allen, R. and Kennedy, K.: Optimizing Compilers for Modern Architectures, *Morgan Kaufmann Publisher* (2002).
9) 木村啓二, 加藤孝幸, 笠原博徳: 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, 情報処理学会論文誌, Vol. 42, No. 4 (2001).
10) 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).
11) 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).
12) 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).
13) *CoSy compiler development system*. http://www.ace.nl/compiler/cosy.html.
14) *ISO/IEC 9899:1999 - Programming Language C* (1999).
15) 白子準, 吉田宗弘, 押山直人, 和田康孝, 中野啓史, 鹿野裕明, 木村啓二, 笠原博徳: マルチコアプロセッサにおけるコンパイラ制御低消費電力化手法, 情報処理学会論文誌, Vol. 47, No. ACS15 (2006).
16) Lee, C., Potkonjak, M. and Mangione-Smith, W. H.: MediaBench: a tool for evaluating and synthesizing multimedia andcommunications systems, *In 30th Annual IEEE/ACM International Symposium on Microarchitecture* (1997).
17) (財)NHK エンジニアリングサービス: DVD 版システム評価用標準動画像シリーズ I.
18) *UZURA3:MPEG1/LayerIII Encoder in FORTRAN90*. http://members.at.infoseek.co.jp/kitaurawa/index_e.html.
19) Guthaus, M. R., Ringenberg, J. S., Ernst, D., Austin, T. M., Mudge, T. and Brown, R. B.: MiBench: A free, commercially representative embedded benchmark suite, *IEEE 4th Annual Workshop on Workload Characterization* (2001).
20) 小高剛, 中野啓史, 木村啓二, 笠原博徳: チップマルチプロセッサ上での MPEG2 エンコードの並列処理, 情報処理学会論文誌, Vol. 46, No. 9 (2005).