Vol.2010-ARC-188 No.9
2010/3/1

組込み向けマルチコア上での複数アプリケーション動作時の自動並列化されたアプリケーションの処理性能

宮本孝道[†1] 間瀬正啓[†1] 木村啓二[†1]
石坂一久[†2] 酒井淳嗣[†2]
枝廣正人[†2] 笠原博徳[†1]

組込み向けマルチコアではユーザの入力などにより複数の逐次あるいは並列プロセスが動作される環境においても高い性能を得ることが重要となる．複数のアプリケーションが同時に実行される環境では，性能低下への対策として共有リソースの競合を減少させることが重要となる．本論文では，アプリケーションの複数同時実行時の OSCAR 自動並列化コンパイラにより生成されたプログラムの並列処理性能を NEC エレクトロニクス NaviEngine 上で評価した．コンパイラにより最適化された MPEG2 デコードと他アプリケーションを同時実行した場合には MPEG2 デコードは最大で 0.91%の性能低下に抑えられ，SPEC95 CFP 101.tomcatv ではコンパイラによる複数のキャッシュ最適化コードを同時実行した場合においても最大で 1.06%の性能低下に抑えられ，性能低下が起こらないことが確かめられた．

Processing Performance of Automatically Parallelized Applications on Embedded Multicore with Running Multiple Applications

Takamichi Miyamoto,[†1] Masayoshi Mase,[†1]
Keiji Kimura,[†1] Kazuhisa Ishizaka ,[†2] Junji Sakai ,[†2]
Masato Edahiro [†2] and Hironori Kasahara [†1]

On embedded multicores, it is important which high performance is obtained although multiple sequential or parallel applications run together. However, performance degradation is occurred by competing resources of multicores. In this paper, we have evaluated parallel performance of programs generated by OSCAR automatic parallelizing compiler in an environment where multiple applications run on NaviEngine developed by NEC Electronics Corporation. When a MPEG2 decoder and other application run together, a MPEG2 decoder's performance degradation is little, a maximum of 0.91% performance degradation. When some SPEC95 CFP 101.tomcatv with cache optimizations by OSCAR automatic parallelizing compiler run together, it is verified which performance degradation is little, a maximum of 1.06% performance degradation.

1. はじめに

近年，情報家電向け組込み分野におけるマルチコアの採用が進み，SCE・IBM・東芝の Cell Broadband Engine[1)]，ARM の MPCore[2)]，NEC エレクトロニクスの NaviEngine[3)]，富士通の FR1000[4)]，パナソニックの UniPhier[5)]，ルネサステクノロジの SH-X3[6)] が発表され，携帯機器，カーナビ，デジタルテレビやゲーム機等に利用され始めている．これらのマルチコア上ではアプリケーションの単独動作のみならず，複数アプリケーションの同時実行が考えられる．特に，並列化されたメディアコーデックのアプリケーションを逐次プログラムとして記述された GUI から操作する等，並列アプリケーションを含めた複数のアプリケーションを連携させて運用する状況が多くなると予想される．

並列アプリケーションの実行時や複数アプリケーションの同時実行時においては，複数コアからの共有リソース利用はリソース競合による性能低下を招く懸念がある．同一システム上で他アプリケーションの実行を想定せずにスケジューリングされた並列アプリケーションでは，性能に対する影響がさらに大きいと予想される．

High Performance Computing（HPC）分野やデスクトップ PC においては共有の L2 および L3 キャッシュを持つマルチコアが多く用いられている．これらのマルチコアでは共有リソースとして共有キャッシュが存在し，このようなメモリアーキテクチャを前提とした複数プロセス動作時の共有キャッシュの分割方式が研究されている[7)]．

一方，組込み向けマルチコアでは，ARM MPCore を搭載した NEC エレクトロニクスの NaviEngine のような L1 キャッシュのみを持ち，複数コア共有キャッシュを持たないアーキテクチャも少なくない．今回対象とする NEC エレクトロニクス開発 NaviEngine では共有

†1 早稲田大学
Waseda University
†2 日本電気株式会社
NEC Corporation

リソースは共有メモリ・バス・I/O が挙げられ，メモリ・バスに対しては L1 キャッシュメモリへの最適化によって共有リソースへのアクセス自体を減少させることができると考えられる．

我々が従来より開発している OSCAR 自動並列化コンパイラでは，キャッシュおよびローカルメモリ最適化手法であるデータローカライゼーション手法[8] を有している．データローカライゼーション手法により，各コアの L1 キャッシュヒット率が最適化され，共有リソースであるバスやメインメモリアクセスが最小化できると考えられる．本論文では，OSCAR 自動並列化コンパイラにより最適化されたアプリケーションの単独動作時の処理性能，および複数アプリケーション同時実行時の処理性能を評価する．

以下本論文では，第 2 章で OSCAR 自動並列化コンパイラによる並列処理手法について，第 3 章で複数アプリケーション同時実行時の評価環境について，第 4 章で性能評価結果について，第 5 章では本論文のまとめを述べる．

2. OSCAR 自動並列化コンパイラによる粗粒度タスク並列処理

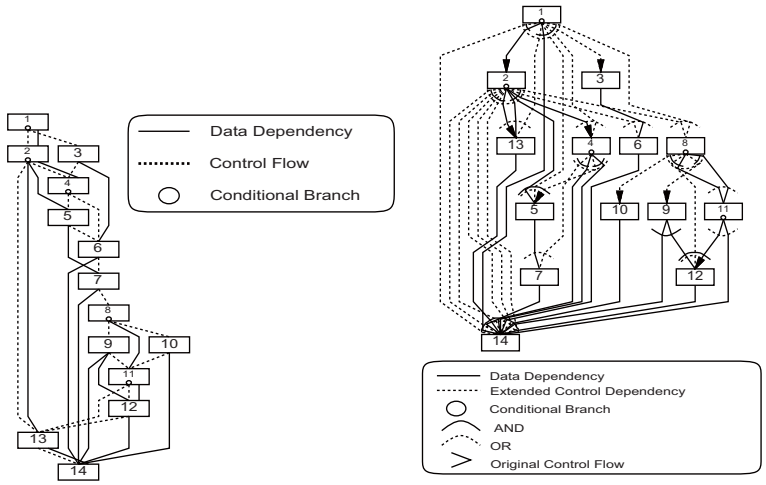
本章では，粗粒度タスク並列性の抽出，データローカリティ最適化手法であるデータローカライゼーション手法，粗粒度タスクスケジューリング手法について述べる．

2.1 粗粒度タスク並列性抽出

粗粒度タスク並列処理では，まずソースプログラムを基本ブロックあるいは基本ブロックを融合・分割した形である疑似代入文ブロック（BPA），ループの一般形であ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）に分割する．ループ並列処理不可能な実行時間の大きい RB や，インライン展開を効果的に適用できない SB に対しては，その内部を階層的に粗粒度タスクに分割して並列処理を行う．

マクロタスクの生成後，マクロタスク間のコントロールフローとデータ依存を解析し，**図 1**（a）に示すようなマクロフローグラフ（MFG）を生成する．図 1（a）の各ノードはマクロタスクを表し，実線エッジはデータ依存を，点線エッジはコントロールフローを表す．また，ノード内の小円は条件分岐を表す．

MFG 生成後，コンパイラは並列性を抽出するためにコントロールフローとデータ依存の両方を考慮した最早実行可能条件解析を MFG に対して行う．マクロタスクの最早実行可能条件とは，コントロール依存とデータ依存を考慮したそのマクロタスクが最も早い時点で実行可能になる条件である．マクロタスクの最早実行可能条件は図 1（b）に示すようなマクロタスクグラフ（MTG）で表される．MFG，MTG 共にエッジの矢印は省略されているが，下向きが想定されている．MTG では横に並んでいるマクロタスク間の並列性が表現されている．



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

図 1 マクロフローグラフとマクロタスクグラフ

2.2 データローカライゼーション手法

データローカライゼーション手法とは，アクセス速度の異なる複数のメモリ機構を持つアーキテクチャにおいて，プロセッサ近傍の高速なメモリを有効に利用して，処理速度の向上を図るものである．データローカライゼーション手法は従来の単一ループネスト内のブロッキングを用いたキャッシュ最適化とは異なる，複数のループネスト集合に対するグローバルなキャッシュあるいはローカルメモリ最適化技術である．データ依存を持つ複数のループをそれらの使用データサイズがキャッシュあるいはローカルメモリサイズにおさまるように，複数ループを整合して分割をする[8]．分割後のデータローカライザブルグループ（DLG）を同一プロセッサで連続実行することでキャッシュあるいはローカルメモリを介したデータの授受により複数ループネスト間でのデータ転送の最小化が可能となる．

2.3 粗粒度タスクスケジューリング手法

粗粒度タスク並列処理では，生成されたマクロタスクを複数のプロセッサエレメント（PE）

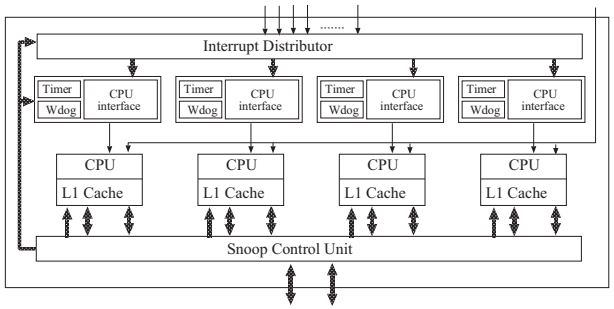


図 2 ARM11 MPCore ブロック図概要

から構成される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．本論文では MTG がデータ依存エッジのみを持つ場合にコンパイラがスタティックにマクロタスクの PG への割り当てを決定することで実行時スケジューリングオーバーヘッドを無くし，データ転送と同期のオーバーヘッドを最小化することが可能であるスタティックスケジューリングを用いた．本論文では ETF/CP（Earliest Task First/ Critical Path）法に対して DLG に属するマクロタスクを同一の PG への連続割り当てを優先したマルチプロセッサスケジューリングアルゴリズム ETF/CP considering DLG 法を適用した．本論文では中粒度，近細粒度の並列性を利用しないため以後 PG を PE として表す．

3. 複数アプリケーション同時実行における評価環境

本章では，複数アプリケーションの同時実行時の評価環境として，NEC エレクトロニクス開発 NaviEngine，複数アプリケーションの動作環境，および評価対象アプリケーションについて述べる．

3.1 NEC エレクトロニクス開発 NaviEngine

NEC エレクトロニクス開発の NaviEngine は 4 コア ARM11 MPCore を搭載している．ARM11 MPCore マルチコアは**図 2** に示すように，ARM11 コアを 4 基集積のホモジニアスマルチコアであり，各コアは 32KB のデータキャッシュ，32KB の命令キャッシュを有する．L1 キャッシュは Snoop Control Unit（SCU）によりデータ一貫性が保証される．本論文で用いた NaviEngine の評価環境概要を**表 1** に示す．

表 1 NaviEngine 評価環境概要

プロセッサコア	ARM11 399MHz
L1 命令キャッシュ	32KB（4way）
L1 データキャッシュ	32KB（4way）
OS	Linux（kernel 2.6.21.6_arm1）
C コンパイラ	GNU gcc-4.2.4
FORTRAN コンパイラ	GNU gfortran-4.4.0

3.2 複数アプリケーション動作環境

本章では，複数アプリケーションの同時実行環境の概要を示す．

本論文では，並列化アプリケーションを含む複数のアプリケーションを連携させて実行する状況，および特に最適化を行われない場合はメモリシステムに対する負荷が大きいアプリケーションを複数実行する状況の二種類の状況を想定して評価を行う．アプリケーション間のデータ共有，および複数アプリケーションを用いた実機による評価の容易性を考慮し，今回は 1 つのプロセス内部で pthread により複数スレッドを起動し，これらのスレッドに各アプリケーションのスレッドをマッピングするという方法で評価を行った．なお，起動するスレッドの数は NaviEngine のコア数を超えないものとする．

図 3 にプログラム全体のメイン関数から呼び出されるアプリケーション A とアプリケーション B の二種類の 2 コア用に並列化されたアプリケーションの 4 コアマルチコアへのスレッドのコア割り当てと動作イメージを示す．プログラム全体のメイン関数からアプリケーション B のメイン関数が pthread により起動され，アプリケーション A は全体のメイン関数から呼び出されることで，二種類のアプリケーションはそれぞれ別スレッドで実行される．また，各アプリケーション内では並列処理時の残りのスレッドが呼び出される．本評価では，スレッド実行コアが重複しないように Linux kernel 2.5.8 から導入されたプロセスが実行を許可される CPU 集合を設定する “sched_setaffinity” を用い，各スレッドの実行コアを固定させる．

3.3 評価対象アプリケーション

本論文では，評価対象アプリケーションとして，マルチメディア処理から MPEG2 デコードと科学技術計算から SPEC95 CFP 101.tomcatv を用いた．

3.3.1 MPEG2 デコード

MPEG2 デコード処理は，可変長複合化，逆量子化，逆量子化後の各係数値の制限処理，逆離散コサイン変換，動き補償予測，足し合わせ処理の 6 つのステージからなる．MPEG2 デコード処理ではスライスレベルの並列性とスライス処理内部でのマクロブロックレベルの

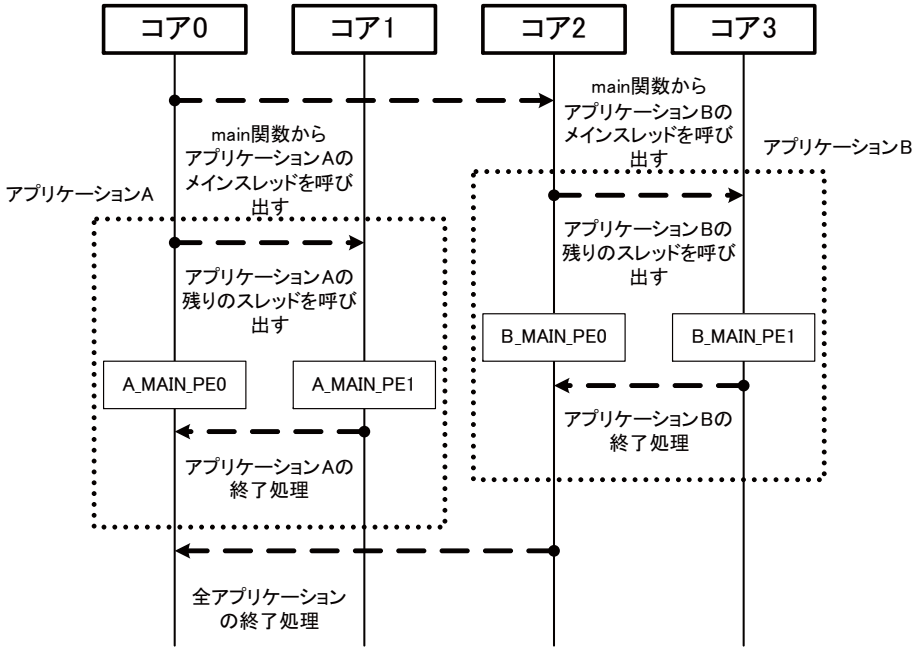


図 3 複数アプリケーション同時実行処理イメージ

並列性が存在する．本論文では並列性向上のためにスライスに対する可変長復号化処理中のスライスヘッダの検出処理を分割するプレスキャニング手法を適用した[9]．プレスキャニング手法ではビットストリームを先頭から走査するためにスライス毎に逐次で処理を行う必要がある．本論文では，MediaBench[10] の “mpeg2decode” を Parallelizable C[11] により参照実装した MPEG2 デコードプログラムを使用した．本プログラムは，主に複数アプリケーションを連携させて実行する状況の評価に用いる．

3.3.2 SPEC95 CFP 101.tomcatv

tomcatv は SPEC95 CFP に含まれるプログラムの 1 つで，ベクトル化メッシュを生成するプログラムであり，サブルーチンや関数を持たず，実行時間の 99%以上を占める収束ループを持つ．このループにはイタレーション間の並列性がないが，内部は BB と並列処理可能ループから構成されている．

4. 複数アプリケーション動作時の並列性能評価

本章では OSCAR 自動並列化コンパイラにより最適化された評価対象アプリケーションの NEC エレクトロニクス開発 NaviEngine 上での並列処理性能について述べる．

4.1 MPEG2 デコードと画面表示アプリケーションの同時実行

画像表示を行うアプリケーションは，Simple DirectMedia Layer（SDL）[12] を用い，共有メモリ上にある YUV データの描写を行う．画像表示アプリケーションと MPEG2 デコードの接続は第 3 章の環境においてコア 0 で実行されるアプリケーション A が画像表示アプリケーション，コア 1 からコア 3 で実行される逐次および並列アプリケーション B が MPEG2 デコードにあたる．また，画像表示アプリケーションと MPEG2 デコードの間でのコア間同期機構，データの受渡しは共有メモリを用いて実現する．本評価では，MPEG2 デコードの入力データとして QVGA（320x240）100 フレームの MPEG2 圧縮データを用いた．また，画像表示は毎秒 15 フレームでの表示を行っている．

MPEG2 デコードの並列処理性能を**図 4** に示す．図 4 において，横軸に単独動作時の 1PE から 3PE（“alone”）と画像表示アプリケーションとの接続時の 1PE から 3PE（“w/ display”）を示し，縦軸に単独動作時の MPEG2 デコードの逐次処理に対する速度向上率を示す．図 4 では各バーはそれぞれの環境における MPEG2 デコードの単独動作時の逐次処理に対する速度向上率を示し，速度向上の値の下には画像表示との接続時の単独動作時に対する性能低下率を示す．

図 4 では，単独動作時の 1PE に対して，2PE で 1.85 倍，3PE で 2.34 倍の並列性能が得られ，また，画像表示との接続時においても，2PE で 1.85 倍，3PE で 2.32 倍の並列性能が得られた．特に，3 コア利用時では，単独動作時に対して画像表示との接続を行った場合でも最大 0.66%の性能低下に抑えられており，ほとんど性能低下を起こしていないことが確かめられた．

4.2 MPEG2 デコードと I/O 実行アプリケーションの同時実行

同様に，コア 0 で実行されるアプリケーション A としてファイルリード・ファイルライトを繰り返し行い，その頻度を調整することにより共有リソースへのアクセスの状況を変更して評価を行った．本アプリケーションは，コア 1 からコア 3 で並列実行されるアプリケーションの実行終了まで，無限ループ内部でファイルリード・ファイルライト処理と共有リソースへのアクセスが無い演算処理を数 100 ミリ秒単位での実行割合（0 から 100 のパーセント記述）を満たすよう繰り返すアプリケーションとなっている．本評価では，コア 1 か

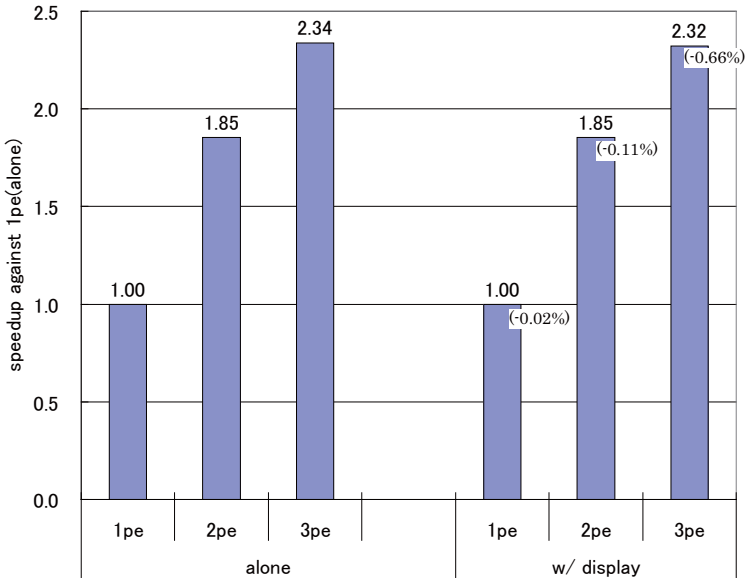


図 4 画像表示と接続時の MPEG2 デコード並列化評価

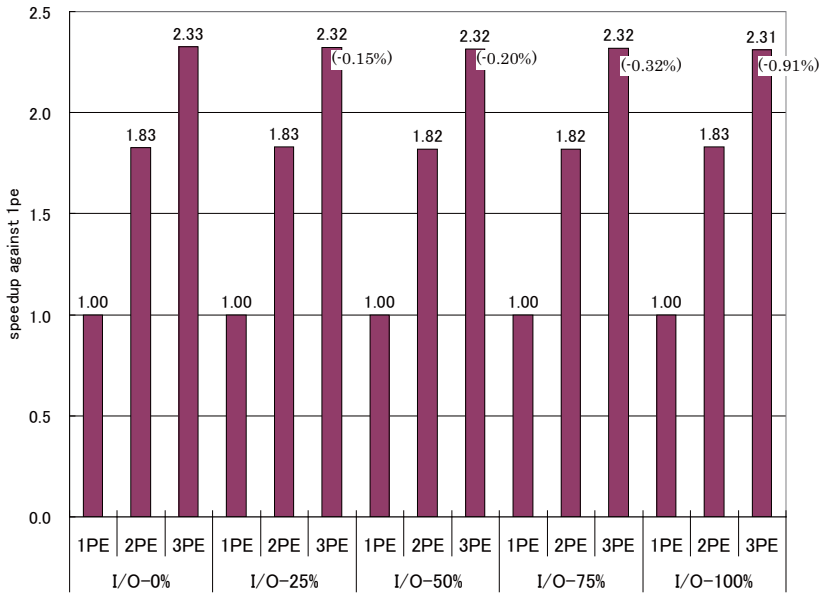


図 5 I/O 実行割合を変化させた時の MPEG2 デコード並列性能

らコア 3 で MPEG2 デコードの処理を行い，入力データとして QVGA（320x240）100 フレームの MPEG2 圧縮データを用いた．

MPEG2 デコードの I/O 実行割合を 0%，25%，50%，75%，100%と変化させた時の並列性能評価結果を**図 5** に示す．図 5 において，横軸は I/O 割合を変化させた時（0%，25%，50%，75%，100%）の処理コア数，縦軸に同一 I/O 割合時の I/O アプリケーションと同時実行させた場合の MPEG2 デコードの逐次処理に対する速度向上を示す．

図 5 より，逐次処理に対する 3 コア使用時の速度向上率では I/O 実行割合が 0%の時の 2.33 倍，I/O 実行割合が 25%の時の 2.32 倍，I/O 実行割合が 50%の時の 2.32 倍，I/O 実行割合が 75%の時の 2.32 倍，I/O 実行割合が 100%の時の 2.31 倍と，I/O の実行割合を変化させた時においても並列性能が変わらないことが分かる．MPEG2 デコードの 3 コア利用時において，I/O 実行割合 0%時に対する I/O 実行割合 100%時では性能低下は 0.91%に抑えられた．これは，並列化された MPEG2 デコードにおいてメモリ・バスの共有リソースへのアクセスが抑えられていることが考えられる．

4.3 SPEC95 CFP 101.tomcatv の複数同時実行

tomcatv のオリジナルコードと OSCAR 自動並列化コンパイラの生成したコードの単独実行および複数の tomcatv を同時に実行した場合の性能評価を利用コア数毎に行った．入力データとして，SPEC95 付属の “ref” セットを用い，入力および出力の時間は性能評価から除外している．

4.3.1 tomcatv でのキャッシュ最適化効果および並列処理性能

本評価では，tomcatv における OSCAR 自動並列化コンパイラのキャッシュ最適化効果および並列処理性能を評価する．

tomcatv を単独で実行させた場合の並列性能評価結果を**図 6** に示す．図 6 の各バーは左からオリジナルコードの逐次処理を 1 コアのみで実行したもの（“original”），OSCAR 自動並列化コンパイラにより最適化された逐次処理を 1 コアのみで実行したもの（“oscar_1pe”），OSCAR 自動並列化コンパイラにより最適化された 2 コア用並列処理を 2 コアのみで実行

表 2　tomcatv の L1 キャッシュと外部メモリリクエスト評価結果

計測項目	original	oscar_1pe
参照ミス回数	13.53×10^8	6.37×10^8
定義ミス回数	3.22×10^8	2.66×10^8
参照ミス率 (%)	7.12%	3.37%
定義ミス率 (%)	5.07%	3.93%
参照ミス回数比 [oscar_1pe / original](%)	47.07%	
定義ミス回数比 [oscar_1pe / original](%)	82.83%	
外部メモリリクエスト回数比 [oscar_1pe / original](%)	55.28%	

したもの（“oscar_2pe”），OSCAR 自動並列化コンパイラにより最適化された 3 コア用並列処理を 3 コアで実行したもの（“oscar_3pe”），OSCAR 自動並列化コンパイラにより最適化された 4 コア用並列処理を 4 コアで実行したもの（“oscar_4pe”）を示している．図 6 では，縦軸にはオリジナルコードの逐次処理（“original”）に対するそれぞれの評価の速度向上を示す．また，**表 2** にオリジナルコードと OSCAR 自動並列化コンパイラによるキャッシュ最適化を適用した逐次処理コード実行時における L1 キャッシュのミス回数およびミス率の結果と，オリジナルコードに対する OSCAR 自動並列化コンパイラによるキャッシュ最適化を適用した逐次処理コードの L1 キャッシュのミス回数の比率と外部メモリへのリクエスト回数比率を示す．

これらの結果から，OSCAR 自動並列化コンパイラにより最適化されたコードでは，L1 キャッシュのミス回数が参照ミスが 49.70%，定義ミスが 82.83%と削減され，外部メモリへのリクエスト回数が 55.28%減少と，共有リソースへのアクセスを削減させたことが分かる．これにより，オリジナルコードに対して逐次処理で 1.38 倍，並列処理の 2 コア利用時で 3.31 倍，3 コア利用時で 5.01 倍，4 コア利用時で 6.57 倍の速度向上が得られた．

4.3.2　tomcatv での複数の逐次処理を同時実行時の処理性能

オリジナルコードの逐次処理と OSCAR 自動並列化コンパイラにより最適化された逐次処理をそれぞれ複数同時実行した場合の評価結果を**図 7** に示す．図 7 の各バーは，左からオリジナルコードの逐次処理を 2 つ同時に 2 コアを使って実行したもの（“original_x2”），OSCAR 自動並列化コンパイラにより最適化された逐次処理を 2 つ同時に 2 コアを使って実行したもの（“oscar1pe_x2”），オリジナルコードの逐次処理を 3 つ同時に 3 コアを使って実行したもの（“original_x3”），OSCAR 自動並列化コンパイラにより最適化された逐次処理を 3 つ同時に 3 コアを使って実行したもの（“oscar1pe_x3”），オリジナルコードの逐次処理を 4 つ同時に 4 コアを使って実行したもの（“original_x4”），OSCAR 自動並列化コンパイラにより

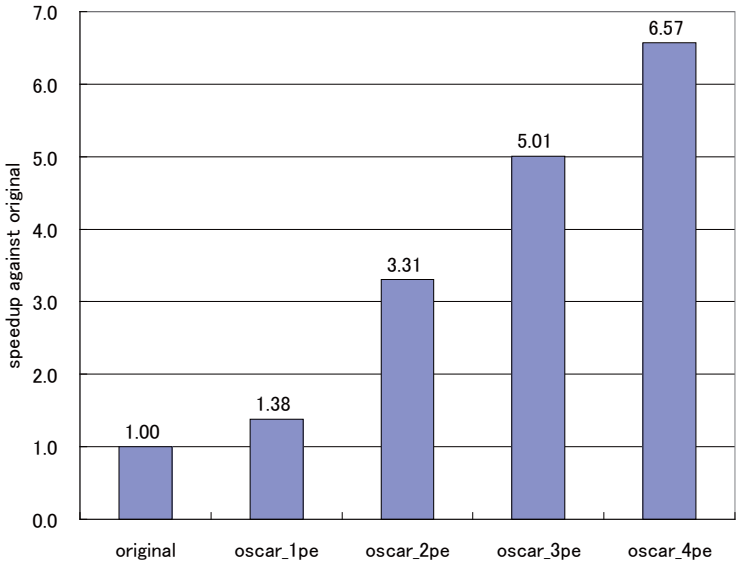


図 6　tomcatv の並列処理性能

最適化された逐次処理を 4 つ同時に 4 コアを使って実行したもの（“oscar1pe_x4”）をそれぞれ示す．図 7 では，横軸にはそれぞれの評価で用いたコア数（“2cores” から “4cores”），縦軸にはオリジナルコードの逐次処理（図 6 の “original”）に対するそれぞれの評価の速度向上を示す．また，各バーの速度向上値とともにに図 6 における評価結果に対する性能低下率を示す．本評価では同時実行されたアプリケーションの内，最も実行が遅かったものを評価結果として採用している．

本評価結果では，オリジナルコードの逐次処理を複数同時実行した場合の性能低下率はそれぞれ 2 つ同時実行時に 0.62%，3 つ同時実行時に 2.00%，4 つ同時実行時に 3.26%，OSCAR 自動並列化コンパイラにより最適化された逐次処理を複数同時実行した場合の性能低下率はそれぞれ 2 つ同時実行時に 0.10%，3 つ同時実行時に 1.00%，4 つ同時実行時に 1.48%であった．特に，オリジナルコードを同時実行した場合と OSCAR 自動並列化コンパイラによるキャッシュ最適化を適用した逐次処理コードを同時実行した場合では，4 コア利用時の複数アプリケーションの同時実行においてオリジナルコードを 4 つ同時実行した

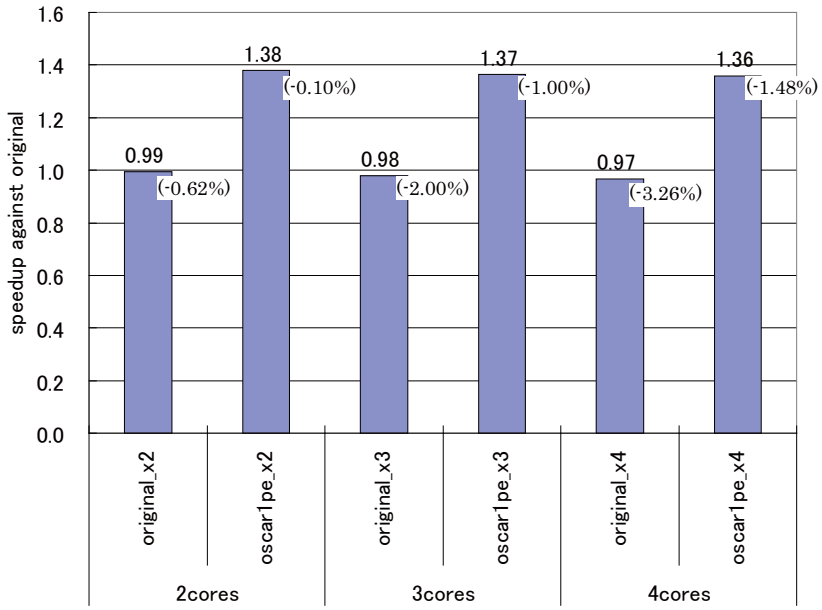


図 7 tomcatv の複数同時実行時の処理性能

場合には 3.26%の性能低下を示していたのに対し，OSCAR 自動並列化コンパイラによるキャッシュ最適化を適用した逐次処理コードを 4 つ同時実行した場合には 1.48%の性能低下に留まる結果が得られ，性能低下を抑えることが確かめられた．

4.3.3 tomcatv での複数の最適化コードを同時実行時の処理性能

OSCAR 自動並列化コンパイラにより最適化された逐次処理および並列処理を同時実行した場合の評価結果を**図 8** に示す．図 8 の各バーは，左から OSCAR 自動並列化コンパイラにより最適化された逐次処理と OSCAR 自動並列化コンパイラにより最適化された 2 コア用並列処理を同時に 3 コアを使って実行した場合（“1pe_2pe”）の逐次処理（“1pe”）と 2 コア用並列処理（“2pe”），OSCAR 自動並列化コンパイラにより最適化された逐次処理と OSCAR 自動並列化コンパイラにより最適化された 3 コア用並列処理を同時に 4 コアを使って実行した場合（“1pe_3pe”）の逐次処理（“1pe”）と 3 コア用並列処理（“3pe”），OSCAR 自動並列化コンパイラにより最適化された逐次処理を二つと OSCAR 自動並列化コンパイラにより最適化された 2 コア用並列処理を同時に 4 コアを使って実行した場合（“1pe_1pe_2pe”）の逐次処理（“1pe_A”）と逐次処理（“1pe_B”）と 2 コア用並列処理（“2pe”），OSCAR 自動並列化コンパイラにより最適化された 2 コア用並列処理を 2 つ同時に 4 コアを使って実行した場合（“2pe_2pe”）の 2 コア用並列処理（“2pe_A”）と 2 コア用並列処理（“2pe_B”）をそれぞれ示す．図 8 では，横軸にはそれぞれの評価で用いたコア数（“3cores” および “4cores”），縦軸にはオリジナルコードの逐次処理（図 6 の “original”）に対するそれぞれの評価の速度向上を示す．また，各バーの速度向上値とともにに図 6 における OSCAR 自動並列化コンパイラにより最適化された逐次処理および並列処理に対する性能低下率を示す．

本評価結果では，OSCAR 自動並列化コンパイラにより最適化されたコードの各性能低下率は逐次処理と 2 コア用並列処理の同時実行時には 1pe 側は 0.47%，2pe 側は 0.34%の性能低下，逐次処理と 3 コア用並列処理の同時実行時には 1pe 側は 0.48%，3pe 側は 0.67%の性能低下，逐次処理を 2 つと 2 コア用並列処理の同時実行時には 1pe_A 側は 0.43%，1pe_B 側は 0.52%，2pe 側は 0.93%の性能低下，2 コア用並列処理を 2 つの同時実行時には 2pe_A 側は 0.95%，2pe_B 側は 1.06%の性能低下となり，OSCAR 自動並列化コンパイラにより最適化された逐次処理および並列処理を複数同時実行させた場合においても，最大で 1.06%の性能低下とほとんど性能低下を起こさないということが確かめられた．

5. ま と め

本論文では，キャッシュ最適化を有する OSCAR 自動並列化コンパイラの MPCore 搭載 NEC エレクトロニクス開発 NaviEngine 上での OSCAR 自動並列化コンパイラにより並列化されたアプリケーションの単独動作時の並列処理性能評価，および複数アプリケーションの同時実行時の並列処理性能評価を行った．複数アプリケーション同時実行環境において，画像表示アプリケーションと MEPG2 デコードの同時実行時では MPEG2 デコードは単独動作時に対して最大で 0.66%の性能低下に留まった．また，I/O 実行割合を 0%から 100%まで変化させた場合の I/O 実行アプリケーションと MPEG2 デコードの同時実行時では，MPEG2 デコードの 3 コアでの並列処理性能は 2.33 倍から 2.31 倍と I/O 実行割合にかかわらず並列処理性能を得ることができ，I/O 負荷を 100%とした場合の性能低下は最大 0.91%に抑えられた．そして，SPEC95 CFP 101.tomcatv では，逐次コードを 4 つ同時実行した場合においてオリジナルコードでは 3.26% の性能低下が，キャッシュ最適化を適用したコードでは 1.48% に抑えることができた．また，複数の OSCAR 自動並列化コンパイラにより最適化された tomcatv を同時に実行した場合では最大で 1.06%の性能低下と，ほ

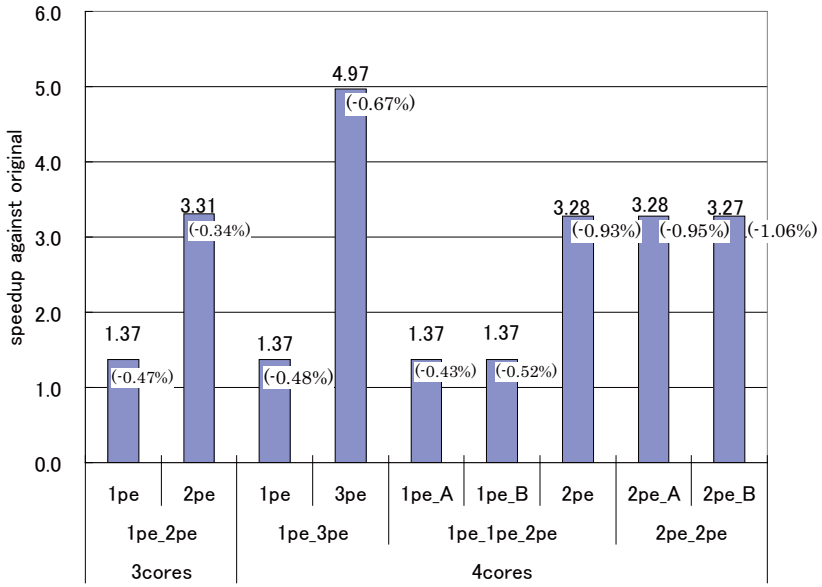


図 8 tomcatv の OSCAR 自動並列化コンパイラによる最適化コード同時実行時の処理性能

とんど性能低下を起こさないことが確かめられた.

NEC エレクトロニクス開発 NaviEngine において，OSCAR 自動並列化コンパイラにより最適化されたアプリケーションは単独動作時に高い並列性能を得ることができ，また，複数アプリケーションの同時実行時においてもキャッシュ最適化による主記憶（オフチップ共有メモリ）アクセスの抑制によりほとんど性能低下が起きず，別プロセスによる負荷に依存せずに並列性能を得ることを確認できた.

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” で開発された OSCAR 自動並列化コンパイラ，および OSCAR API を利用して行われた.

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) NEC エレクトロニクス株式会社：NaviEngine. http://www.necel.com/automotive/ja/assp/naviengine.html.
4) Suga, A. and Imai, S.: FR-V Single-Chip Multicore Processor:FR1000, *Fujitsu Sci Tech J*, Vol.42, No.2, pp.190–199 (2006).
5) 木村浩三ほか：デジタル家電統合プラットフォーム UniPhier におけるメディアプロセッサ，DA シンポジウム (2005).
6) Kamei, T.: SH-X3 : An Enhanced SuperH Core for Low-power MP Systems, *Fall Microprocessor Forum 2006* (2006).
7) Iyer, R.: CQoS: a framework for enabling QoS in shared caches of CMP platforms, *ICS '04: Proceedings of the 18th annual international conference on Supercomputing*, New York, NY, USA, ACM, pp.257–266 (2004).
8) 石坂一久ほか：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).
9) Iwata, E. and Olukotun, K.: Exploiting Coarse-Grain Parallelism in the MPEG-2 Algorithm, *Technical Report CSL-TR-98-771* (1998).
10) C. Lee et al.: MediaBench: A Tool for Evaluating and Synthesizing Multimedia and Communications Systems, *30th International Symposium on Microarchitecture (MICRO-30)* (1997).
11) 間瀬正啓ほか：マルチコアにおける Parallelizable C プログラムの自動並列化，情処処理学会研究報告， Vol.2009-ARC-184, No.15, pp.1–10 (2009).
12) Sam Lantinga: Simple DirectMedia Layer. http://www.libsdl.org/.