OSCARチップマルチプロセッサ上でのマルチグレイン並列処理

木村啓二[†]　小高　剛[††]
小幡元樹[†,†††]　笠原博徳[††,†††]

あらまし　本論文では，コンパイラ協調動作型チップマルチプロセッサ OSCAR Chip Multiprocessor (OSCAR CMP) 上でのマルチグレイン並列処理について述べる．OSCAR CMP は，チップ上トランジスタの有効利用によるスケーラブルな性能向上と，コンパイラサポートによるプログラム開発効率の向上を目的として研究・開発されているチップマルチプロセッサアーキテクチャである．本目的を達成するため OSCAR CMP は，プログラム実行文レベルの並列性を利用する近細粒度並列処理に，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，及びループやサブルーチン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の利用を前提に設計されており，プロセッサ内部に簡素な 1 命令発行の CPU コアを複数搭載し，各 CPU はプロセッサプライベートデータ用のローカルメモリ，データローカリティ最適化用の 2 ポートメモリ構成の分散共有メモリ，及びデータ転送最適化用のデータ転送ユニットを持つ．本論文では，SPEC fp 2000/95 ベンチマークにマルチグレイン並列処理を適用し，OSCAR CMP 上で評価した結果を報告する．評価の結果，microSPARC 相当の単一命令発行 CPU コアを 4 基搭載した OSCAR CMP は逐次実行に対して，HYDRO2D で 2.98 倍，TOMCATV で 3.84 倍，MGRID で 3.84 倍，SWIM で 3.97 倍，FPPPP で 2.36 倍，TURB3D で 2.88 倍，SU2COR で 2.64 倍，APPLU で 2.29 倍，APSI で 1.77 倍の速度向上を得ることができ，CPU コアの増加に応じたスケーラブルな性能向を得られることが確認できた．

Multigrain Parallel Processing on OSCAR Chip Multiprocessor

Keiji Kimura[†]　, Takeshi Kodaka[††]　, Motoki Obata[†,†††] and Hironori Kasahara[††,†††]

Abstract　This paper describes multigrain parallel processing on OSCAR Chip Multiprocessor (OSCAR CMP). The aim of OSCAR CMP is to achieve both of scalable performance improvement with effective use of huge number of transistors on a chip and high efficiency of application development with compiler supports. OSCAR CMP integrates simple single issue processors having local data memory for private data recognized by compiler, distributed shared data memory for optimal use of data locality over different loops. The compiler controllable data transfer unit for overlapping data transfer, and the multigrain parallelizing compiler, which exploits statement level near-fine grain parallelism, loop iteration level parallelism and coarse grain task parallelism hierarchically, fully controls these hardwares. Performance of multigrain parallel processing on OSCAR CMP is evaluated using SPEC fp 2000/95 benchmark suite. When microSPARC like single issue core is used, OSCAR CMP having four CPU cores gives us 2.98 times speedup in HYDRO2D, 3.84 times in TOMCATV, 3.84 times in MGRID, 3.97 times in SWIM, 2.36 times in FPPPP, 2.88 times in TURB3D, 2.64 times in SU2COR, 2.29 times in APPLU and 1.77 times in APSI.

1　はじめに

近年の 1 チップ上に搭載できるトランジスタ数の増加に対し，これまでのマイクロプロセッサで主に利用されてきたスーパースカラのような命令レベル並列性のみを用いる方式では，今後大幅な性能向上は困難だと考えられている．そのため，1 プロセッサ内部で命令レベル並列性よりも並列処理粒度の大きいループタレーションレベル並列性やスレッドもしくはプロセスレベルの並列性を利用するプロセッサアーキテクチャが注目を集めている．このようなプロセッサアーキテクチャとして，一つのマイクロプロセッサを擬似的にマルチプロセッサとして利用できる Simultaneous Multi Threading (SMT)[1),2)] や，1 チップ上に複数の CPU コアを搭載する Hydra[3)]，Multiscalar[4)]，SKY[5)]，OchaPRO[6)]，MP98[7)]，Power4[8)] などのチップマルチプロセッサ（CMP）アーキテクチャが提案あるいは製品化されている．特に CMP アーキテクチャは次世代の主要マイクロプロセッサアーキテクチャの一つとして，高性能サーバ用途から，携帯電話，ゲーム機や PDA などの組み込み用途まで採用され始めている．このチップマルチプロセッサの持つ能力を十分に引き出すためには，広域的なプログラム解析技術によりプログラム中の並列性を最大限に抽出可能なコンパイラのサポートが必須である．筆者等は，命令レベルあるいはプログラム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の並列性を利用する中粒度並列処理，及びループやサブルーチン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とチップマルチプロセッサが協調動作することにより，実効性能が高く価格性能比の良いコンピュータシステムを構築することができると考え，ソフトウェア協調動作型 OSCAR（Optimally SCheduled Advanced multiprocessoR）チップマルチプロセッサ（OSCAR CMP）を提案し

† 早稲田大学 理工学総合研究センター
Advanced Research Institute for Science and Engineering, Waseda University
†† 早稲田大学理工学部電気電子情報工学科
Dept. of Electrical, Electronics and Computer Engineering, Waseda University
††† アドバンスト並列化コンパイラプロジェクト
Advanced Parallelizing Compiler Project

ている[9]．本論文は，OSCAR CMP アーキテクチャとマルチグレイン並列処理の協調動作及び，SPEC fp 2000/95 を用いた性能評価について報告する．

以下，2 節でマルチグレイン並列処理について，3 節でマルチグレイン並列処理と OSCAR CMP の協調動作について，4 節で SPEC fp を用いた性能評価についてそれぞれ述べる．

2 マルチグレイン並列処理

本節では，OSCAR CMP が利用する並列処理であるマルチグレイン並列処理技術について述べる．マルチグレイン並列処理[10]とは，ループやサブルーチン等の粗粒度タスク間の並列処理を利用する粗粒度タスク並列処理（マクロデータフロー処理）[11),12)]，ループイタレーションレベルの並列処理である中粒度並列処理，基本ブロック内部のステートメントレベルの並列性を利用する近細粒度並列処理[13)]を階層的に組み合わせて，プログラム全域にわたる並列処理を行なう手法である．このマルチグレイン並列処理は，OSCAR FORTRAN コンパイラに実装されている[14)]．

2.1 粗粒度タスク並列処理（マクロデータフロー処理）

粗粒度タスク並列処理では，ソースとなるプログラムを疑似代入文ブロック（BPA），繰り返しブロック（RB），サブルーチンブロック（SB）の三種類の粗粒度タスク（マクロタスク（MT））[11)]に分割する．MT 生成後，コンパイラは BPA，RB，SB 等の MT 間のコントロールフローとデータ依存を解析し，それらを表したマクロフローグラフ（MFG）[12),15)]を生成する．さらに MFG から MT 間の並列性を最早実行可能条件解析[12),15)]により引きだし，その結果をマクロタスクグラフ（MTG）[12),15)]として表現する．MTG に条件分岐等の実行時不確定性がない場合，プロセッサ間データ転送および同期オーバーヘッドを最小化できるように，コンパイラは MTG 上の MT をスタティックにプロセッサあるいはプロセッサグループ（PG）に割り当て，各プロセッサ用コードを生成する．MTG 中に条件分岐がありダイナミックスケジューリングを用いる場合には，MT のコードに加えスケジューラのコードを生成し，実行時に MT をプロセッサあるいは PG に割り当てる．

SB や RB 内部に粗粒度並列性がある場合，その SB や RB 内部をさらにマクロタスクに分割し，粗粒度タスク並列処理を階層的に適用する[16)]．

2.2 中粒度並列処理

PG に割り当てられた MT が Doall 可能な RB である場合，この RB は PG 内のプロセッシングエレメント（PE）に対して，イタレーションレベルで割り当てられ並列実行される．またこの場合には，各 PE 上のローカルメモリ，あるいは分散共有メモリを有効利用するためのローカライゼーション技術[17)]を利用可能である．

2.3 近細粒度並列処理

PG に割り当てられた MT が，BPA や中粒度並列処理を適用できない RB である場合，それらはステートメントレベルのタスクに分割され，PG 内の PE により並列処理される．

近細粒度並列処理においては，BPA 内のステートメント，もしくは複数のステートメントから構成される疑似代入文を一つの近細粒度タスクとして定義する．コンパイラは，BPA を近細粒度タスクに分割した後，タスク間のデータ依存を解析してタスクグラフを作成する．次に，このタスクグラフ上のタスクを，データ転送・同期オーバーヘッドを考慮して実行時間を最小化できるように各 PE にスタティックにスケジューリングする．

OSCAR FORTRAN 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[15)]の 4 手法を同時に適用し最良のスケジュールを選んでいる．上記のようにスタティックスケジューリングを用いることにより，BPA 内で用いられるデータのローカルメモリ，分散共有メモリ，レジスタへの配置等，データ配置の最適化やデータ転送・同期オーバーヘッドの最小化といった各種最適化が可能になる．

スケジューリング後，コンパイラは PE に割り当てられたタスクに対応する命令列を順番に並べ，データ転送命令や同期命令を必要な箇所に挿入し，各 PE 毎に異なるマシンコードを生成する．近細粒度タスク間の同期にはバージョンナンバー法を用い，同期フラグの受信は受信側 PE のビジーウェイトによって行なわれる．

2.4 マルチグレイン並列処理の実行イメージ

次に，OSCAR CMP 上でのマルチグレイン並列処理の実行イメージについて説明する．マルチグレイン並列処理では，異なる粒度の並列性を階層的に組み合わせて利用しているが，階層的なタスク制御（もしくはスレッド制御）の複雑さを避けるため，OSCAR CMP では各プロセッサが各々の実行コードを独自に持っており，プログラム実行中 fork や join などのスレッド制御を行わない[14)]．本節ではこのような実行モデルについて説明する．

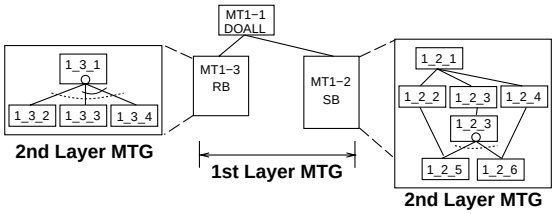
図 1(b) は図 1(a) の 2 階層からなるマクロタスク

グラフを 8 プロセッサで実行したときのイメージを表している．図 1 の例では，条件分岐のない第一階層のマクロタスクグラフにスタティックスケジューリングが適用されている．また，第一階層のマクロタスクグラフでは，高々二つの MT しか並列に実行できないため，8 つのプロセッサが各々4 つのプロセッサを持つプロセッサグループ（PG）に分割され，プロセッサグループ 0（PG0）には MT1_1 と MT1_3，プロセッサグループ 1（PG1）には MT1_2 がスタティックスケジューリングによりそれぞれ割り当てられている．スタティックスケジューリング適用時には，スケジューリング結果にしたがって，コンパイラが各々の PG が実行するマクロタスクのコードを生成する．さらに，割り当てられたマクロタスク内部に粗粒度並列性がある場合は，PG 内の各プロセッサによって階層的な粗粒度並列処理が行われる．本節の例では，MT1_2 と MT1_3 内部の第二階層のマクロタスクグラフが両方とも条件分岐を持っているので，各々のマクロタスクグラフにダイナミックスケジューリングが適用される．

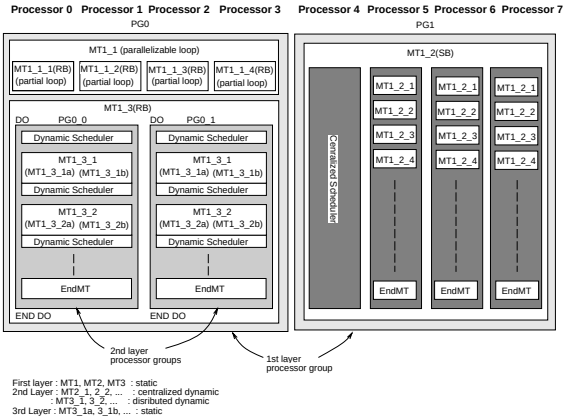
図 1(b) の例では，PG1 に割り当てられた MT1_2 内部のマクロタスクグラフは集中スケジューラ方式によるダイナミックスケジューリングが適用される．本例の場合，プロセッサ 4 が集中スケジューラとして働き，プロセッサ 5–7 で MT1_2 内部のサブマクロタスク MT1_2_1，1_2_2，. . . が集中スケジューラのスケジューリング結果に従い実行される．集中スケジューラ方式では，マクロタスクグラフ内部の各 MT の最早実行可能条件を集中スケジューラが管理し，最早実行可能条件が満たされた MT をレディーキューに投入し，MT の優先順位に従ってアイドルプロセッサに MT を割り当てる．

一方，MT1_3 内部のマクロタスクグラフには分散スケジューリング方式のダイナミックスケジューリングが適用される．本例では MT1_3 内部のサブマクロタスクが，PG0 をさらに二分割して定義された PG0_0 と PG0_1 に割り当てられる．分散スケジューラ方式を適用した MT1_3 内部のサブマクロタスク MT1_3_1，MT1_3_2，. . . は各々がスケジューリングコードを持ち，MT 終了時あるいは条件分岐時に最早実行可能条件の更新を行い，各更新時のスケジューリング結果に従いレディー MT を PG0_0 もしくは PG0_1 に割り当てる．図 1(b) の例では，各々のサブマクロタスクはさらに PG0_0 もしくは PG0_1 内部のプロセッサ 2 基により並列実行される．

本節の例では，マルチグレイン並列処理の実行イメージの説明のために，集中スケジューラ方式と分散スケジューラ方式の二つのダイナミックスケジューリング方式を用いた．基本的にこれらのスケジューリング方式は，利用可能なプロセッサ数と各階層のマクロタスクグラフに内在する並列性を基準として選択される．利用可能なプロセッサ数が少ない場合は，プロセッサ全てをマクロタスクの実行に使用可能な分散スケジューラ方式を選択する．しかしながら分散スケジューラ方式では，各 MT の最早実行可能条件やレディー MT キューなどのスケジューリング情報を共有メモリ領域に配置する必要があり，これらのスケジューリング情報の更新には実行コストの大きい排他制御が必要になる．そのため，利用可能なプロセッサ数が十分多い場合，集中スケジューラが共有情報を管理することにより排他制御が不要となる集中スケジューラ方式を利用することになる．



(a) マクロタスクグラフ



(b) 生成された並列化コードのイメージ

図 1: 8 プロセッサでの実行イメージ

3 マルチグレイン並列処理と OSCAR CMP の協調動作

本節では，マルチグレイン並列処理と OSCAR CMP の協調動作について述べる．まず，OSCAR CMP の概要を説明し，本チップマルチプロセッサアーキテクチャがマルチグレイン並列処理を構成する三種の並列処理をどのようにサポートするか説明する．

3.1 OSCAR CMP のアーキテクチャ

図 2 に，OSCAR CMP を示す．OSCAR CMP は，簡素な CPU 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 ポートメモリ構成の分散共有メモリ（DSM），及びデータ転送ユニット（DTU）を持つプロセッシングユニッ

ト（PE）をチップ内部に複数搭載し，これらの PE を複数バスやクロスバースイッチなどの相互結合網で結合している．

LPM は各々の CPU で実行するプログラムを格納する．DSM は後節で述べるように，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転送及び同期に使用する．LDM はプロセッサのプライベートデータを格納するメモリであり，2 ポート構成の DSM に対し同数のトランジスタ数で 2 倍の容量を確保可能である．また，本 OSCAR CMP はチップ外部に集中共有メモリ（CSM）が接続されている．

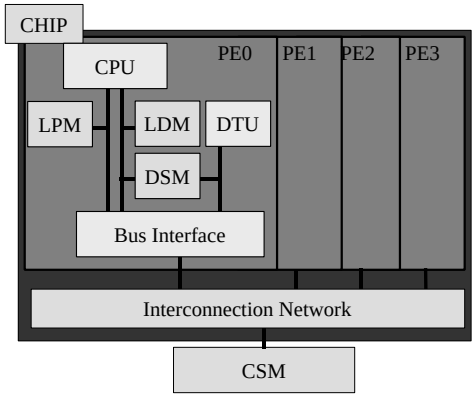


図 2: OSCAR CMP アーキテクチャ

3.2 粗粒度タスク並列処理のサポート

2.1 節で述べたように，マルチグレイン並列処理では，プログラムもしくはプログラムの各部分に内在する粗粒度タスク並列性に応じて，様々な構成のプロセッサグループを定義する必要がある．このようなソフトウェアによるプロセッサグループ構成の柔軟性を確保するため，OSCAR CMP では PE 間相互結合網としてクロスバーネットワークや複数バスなどの，各 PE が等距離で接続されるネットワークを使用する．

あるマクロタスクグラフにスタティックスケジューリングが適用された場合，コンパイラはプロセッサプライベートデータを LDM に，またプロセッサ間共有データを DSM にそれぞれ配置できる．一方，ダイナミックスケジューリングが適用された場合，共有データは CSM に配置される．さらに，分散スケジューリング方式のダイナミックスケジューリングが適用された場合，最早実行可能条件などのスケジューリング情報も CSM に配置される．

スタティックスケジューリング時の MT 間の同期，及びダイナミックスケジューリング時のスケジューリング情報の授受には，DSM を使用する．DSM の使用により，同期フラグのビジーウェイトが PE 間結合網を使用せずに PE 内部で行われるため，効率の良い同期やスケジューリング情報の授受が可能である．

また，RB やループ間で共有されるデータを DSM や LDM の容量を考慮して分割し，データを共有するループを近接して実行するスケジューリングを行うことにより DSM や LDM 経由でデータを授受できるデータローカリティ最適化を適用可能である [18]．さらに，DTU の使用により，残存するデータ転送をプロセッサでのタスク実行とオーバーラップして隠蔽するデータ転送最適化技術が利用可能である [19]．

3.3 中粒度並列処理のサポート

ループイタレーションレベル並列処理では，アレイプライベーティゼーション適用可能な配列を LDM に配置する．また，Doacross やリダクションループにおけるデータ転送は DSM 経由で行われる．

3.4 近細粒度並列処理のサポート

近細粒度並列処理では，コンパイラのスタティックスケジューリング結果どおりに近細粒度タスクが各プロセッサで実行されることが重要である．OSCAR CMP では，コンパイラがパイプラインの挙動を把握可能な，簡素な 1 命令発行のプロセッサを用いることで，近細粒度並列処理をサポートしている．

近細粒度タスク間の同期及びデータ転送に関しては，MT 間の同期と同様に DSM を使用する．先行タスクが別プロセッサに割り当てられた後続タスクに同期フラグやデータを転送する場合，データ転送元のプロセッサは受信先プロセッサの DSM に直接データを書き込む．DSM は 2 ポート構成であり，これらのポートからのアクセスは同時に処理可能なので，受信先プロセッサの処理は妨げられない．また，同期フラグの受信チェックはプロセッサローカルにある DSM に対して行われるので，PE 間結合網のバンド幅を損なうことはない．

4 性能評価

本節では，OSCAR CMP 上でマルチグレイン並列処理を評価した結果について述べる．評価には，SPEC fp 2000/95 を用いた．

4.1 評価環境

本評価で使用した OSCAR CMP では，PE を 1 基から 4 基まで搭載するものとした．また，PE 間結合網には 3 本のバスを使用した．LDM の容量は 256K バイトとし，1 クロックでアクセス可能である．同様に，DSM の容量は 16K バイトで，ローカル DSM アクセスに 1 クロック，リモート DSM アクセスに 4 クロックかかるものとした．本パラメータ使用時では，近細粒度並列処理における同期フラグやデータ転送，及び粗粒度タスク並列処理におけるダイナミックスケジューリングのスケジューリン

グデータの転送に 4 クロックそれぞれかかる．同様に，同期フラグのチェックに 4 クロック，転送されたデータやスケジューリングデータのロードに 1 クロックそれぞれかかる．さらに，CSM のアクセスレイテンシは 20 クロックかかるものとする．

PE 内部の CPU コアには，microSPARC 相当の簡素な 1 命令発行のプロセッサを用いた．この CPU コアは整数演算ユニット（IEU），ロード・ストアユニット（LSU）及び浮動小数点ユニット（FPU）をそれぞれ一つずつ持つ．ただし，本評価で用いた CPU コアの命令セットは SPARC V9 であり，命令実行のレイテンシとスループットは UltraSPARC II と同等である．

評価には，上記の OSCAR CMP を精密に再現できるシミュレータを用いた．

4.2 評価結果

評価では，SPEC CPU fp 2000 及び 95 より 9 本の FORTRAN77 プログラムを評価した．評価時間短縮のため，付録 A 節のようにプログラムの各パラメータを修正した．これらのプログラムに OSCAR マルチグレイン自動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，OSCAR CMP バックエンドを用いて実行バイナリを生成した．ただし，データローカリティ最適化，及びデータ転送ユニットによるデータ転送オーバヘッド隠蔽技術はバックエンドが未対応なので今回の評価には含まれていない．評価結果を図 3 に示す．図は，各アプリケーションにおける，1 プロセッサでの逐次実行に対する 2 プロセッサ及び 4 プロセッサによる並列処理時の速度向上率を表している．

図 3 より 4 プロセッサ使用時に，ループ並列性の高い HYDRO2D で 2.98 倍，TOMCATV で 3.84 倍，MGRID で 3.96 倍，SWIM で 3.97 倍の速度向上を得られていることがわかる．次に，ループ並列性の低さより，従来のコンパイラ及びマルチプロセッサシステムでは十分な性能向上が得られなかった FPPPP で 2.36 倍の速度向上が得られている．これは，FPPPP の実行時間のほとんどを占めるサブルーチン TWLDRV 及び FPPPP 内部の近細粒度並列性を利用しているためである．特に，FPPPP は 333 個の近細粒度タスク（ステートメント）を持ち，このサブルーチンの近細粒度並列処理による性能向上への寄与は大きい．OSCAR CMP では 3.4 節で述べた近細粒度並列処理のアーキテクチャサポートにより，これらのタスクの間の並列性を効率よく利用できる．

SU2COR では，2 プロセッサ使用時で 1.64 倍，4 プロセッサ使用時で 2.64 倍の性能向上をそれぞれ得ている．SU2COR のサブルーチン LOOPS は 130 個の MT からなるマクロタスクグラフから構成され，本評価ではこのマクロタスクグラフを 2 プロセッサ使用時では 2PGx1PE，4 プロセッサ使用時では 2PGx2PE のプロセッサ構成でそれぞれ実行した．ここで，2PGx1PE とは，1 プロセッサからなるプロセッサグループ（PG）二つのプロセッサ構成，2PGx2PE とは 2 プロセッサからなるプロセッサグループ二つのプロセッサ構成であることをそれぞれ表している．

TURB3D においても，2 プロセッサ使用時で 1.98 倍，4 プロセッサ使用時で 2.88 倍の性能向上をそれぞれ得ている．本プログラムのサブルーチン TURB3D も，SU2COR と同様に粗粒度タスク並列性が豊富であり，本評価では 2 プロセッサ及び 4 プロセッサをそれぞれ 2PGx1PE，4PGx1PE 構成で実行した．

APPLU と APSI は他のアプリケーションよりも速度向上率が乏しく，APPLU では 4 プロセッサ使用時で 2.29 倍，APSI では 1.77 倍の速度向上となっている．APPLU にはパイプライン並列化可能なループがあるが，現在の OSCAR FORTRAN コンパイラでは，パイプライン並列性の抽出を行っていないため，内側ループの Doall 並列処理及び近細粒度並列処理のみの結果となっている．また，APSI は多数の Doall ループを持つが，これらのループのループボディと回転数は共に小さく並列性の利用は困難であるため，上記の結果となっている．

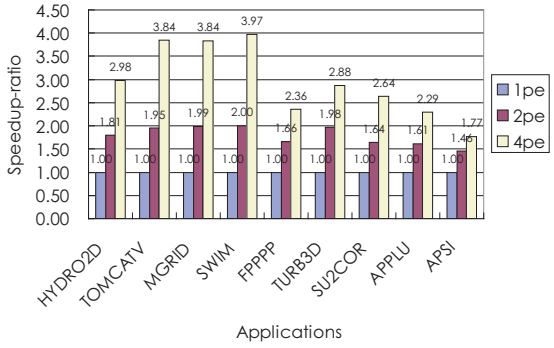


図 3: 評価結果

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について述べ，特に本 OSCAR CMP アーキテクチャのマルチグレイン並列処理サポートについて説明した．また，SPEC fp CPU ベンチマークにマルチグレイン並列処理を適用し，これらのアプリケーションを OSCAR CMP 上で性能評価を行った．評価の結果，microSPARC 相当の簡素な CPU コアを持つ OSCAR CMP は逐次実行に対し，HYDRO2D で 2.98 倍，TOMCATV で 3.84 倍，MGRID で 3.84 倍，SWIM で 3.97 倍，FPPPP で 2.36 倍，TURB3D で 2.88 倍，SU2COR で 2.64 倍，APPLU で 2.29 倍，APSI で 1.77 倍の速度向上

を得ることができ，CPU コアの増加に応じた性能向上を得られることが確認できた．

今後の課題として，データローカリティ最適化及びデータ転送オーバーヘッド隠蔽技術の評価が挙げられる．

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」及び経済産業省ミレニアムプロジェクト「IT21 アドバンスト並列化コンパイラ」により行われた．本論文作成に当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），倉田隆弘氏（ソニー），高山秀一氏（松下）安川英樹氏（東芝）に感謝致します．

参考文献

[1] D. M. Tullsen, S. J. Eggers, and H. M. Levy. Simultaneous multithreding: Maximizing on-chip parallelism. In *Proc. of the 22nd International Symposium on Computer Architecture (ISCA-22)*, June 1995.

[2] D. T. Marr, F. Binns, D. L. Hill, G. Hinton, D. A. Koufaty, J. A. Miller, and M. Upton. Hyper-threading technology architecture and microarchitecture. *Intel Technology Journal*, Vol. 6, No. 1, pp. 1–12, 2001.

[3] L. Hammond, B. Hubbert, M. Siu, M. K. Prabhu, M. Chen, and K. Olukotun. The Stanford HYDRA CMP. *IEEE MICRO Magazine*, Vol. 20, No. 2, pp. 71–84, 2000.

[4] G. S. Sohi, S. E. Breach, and T. N. Vijaykumar. Multiscalar processors. In *Proc. of the 22nd International Symposium on Computer Architecture (ISCA-22)*, 1995.

[5] 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. In *JSPP'98*, June 1998.

[6] 玉造, 平木. Runtime restructuring による複数コントロールフロー予測. 情報処理学会研究報告 2002-ARC-149, August 2002.

[7] NEC Corporation. *MP98 Project*, 2000. http://www.labs.nec.co.jp/ MP98/.

[8] J. M. Tendler, S. Dodson, S. Fields, H. Le, and B. Sinharoy. Power4 system microarchitecture. *Technical White Paper*, Oct 2001.

[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, pp. 1924–1934, May 1999.

[10] H. Kasahara, H. Honda, and S. Narita. A multigrain parallelizing compilation scheme for oscar. In *Proc.4th Workshop on Lang. and Compilers for Parallel Computing*, Aug 1991.

[11] 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.

[12] 本多, 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.

[13] 笠原. マルチプロセッサシステム上での近細粒度並列処理. 情報処理, Vol. 37, No. 7, pp. 651–661, Jul 1996.

[14] 笠原, 小幡, 石坂. 共有メモリマルチプロセッサシステム上での粗粒度タスク並列処理. 情報処理学会論文誌, Vol. 42, No. 4, pp. 910–920, Apr 2001.

[15] 笠原．並列処理技術．コロナ社, 1991.

[16] 岡本, 合田, 宮沢, 本多, 笠原．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．情報処理学会論文誌, Vol. 35, No. 4, pp. 513–521, 1994.

[17] 吉田, 越塚, 岡本, 笠原．階層型粗粒度並列処理における同一階層内ループ間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. 40, No. 5, pp. 2054–2063, May 1999.

[18] H. Kasahara and A. Yoshida. A Data-Localization Compilation Scheme Using Partial Staticc Task Assignment for Fortran Coarse Grain Parallel Processing. *Journal of Parallel Computing*, Vol. Special Issue on Languages and Compilers for Parallel Computers, , May 1998.

[19] H. Kasahara, M. Kogou, T. Tobita, T. Masuda, and T. Tanaka. An automatic coarse grain parallel processing scheme using multiprocessor scheduling algorithm considering overlap of task execution and data transfer. In *Proc. of SCI99 and ISAS*, pp. 82–89, Aug. 1999.

A 評価プログラム

HYDRO2D

本評価では，SPECfp95 の 104.hydro2d を使用した．評価に際して，test データセットを使用し，ソースプログラム中の「MP」及び「NP」をそれぞれ 52 から 20 に変更した．

TOMCATV

本評価では，SPECfp95 の 101.tomcatv を使用した．評価に際して，train データセットを用い，配列サイズ「N」を 257 から 65 に変更した．また，メインループの繰り返し回数「ITACT」を 500 から 200 に変更した．

MGRID

本評価では，SPECfp2000 の 172.mgrid を使用した．評価に際して，test データセットの「LMI」を 7 から 3 に変更した．

SWIM

本評価では，SPECfp2000 の 171.swim を使用した．評価に際して，test データセットのパラメータ「ITMAX」及び「MPRINT」を 10 から 2 に変更した．

FPPPP

本評価では，SPECfp95 の 145.fpppp を使用した．評価に際して ref データセットのパラメータ「NATOMS」を 30 から 2 に変更した．

TURB3D

本評価では，SEPCfp95 の 125.turb3d を使用した．評価に際して，train データセットを用い，パラメータ「NSTEP」を 11 から 6 に,「NAG」を 10 から 5 に，また「IX」「IY」「IZ」をそれぞれ 64 から 16 に変更した．また，粗粒度並列性を解析しやすくするため，いくつかのサブルーチンコール間のデータ依存を手動で解決した．

SU2COR

本評価では，SPECfp95 の 103.su2cor を使用した．評価に際して，test データセットのパラメータ「LSIZE(4)」を 8, 8, 8, 16 からそれぞれ 4, 4, 4, 8 に変更した．

APPLU

本評価では，SPECfp2000 の 173.applu を使用した．評価に際して test データセットを用い，ループ回転数「ITMAX」を 50 から 5 に変更した．

APSI

本評価では，SPECfp95 の 141.apsi を使用した．評価に際して test データセットを用い，パラメータ「NTIME」を 720 から 3 に変更した．