2004－ARC－160　(10)
2004／12／2

OSCAR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

小高　剛[†]　中野啓史[†]
木村啓二[†]　笠原博徳[†]

本論文では，マルチメディアアプリケーションとしてディジタル TV や DVD などのメディアで広く利用されている MPEG2 エンコードに対する，チップマルチプロセッサ上でのメモリ利用最適化およびデータ転送最適化手法を伴う粗粒度タスク並列処理手法の提案を行なうと共に，OSCAR チップマルチプロセッサ上での性能評価を行なう．性能評価の結果，データローカリティの利用およびデータ転送オーバーヘッド隠蔽手法を含む提案する粗粒度タスク並列処理を適用した MPEG2 エンコードは，逐次実行に対し，1 プロセッサ利用時 1.24 倍，2 プロセッサ利用時 2.46 倍，4 プロセッサ利用時 4.57 倍，8 プロセッサ利用時 7.97 倍，16 プロセッサ利用時 11.93 倍の速度向上率が得られることが確認できた．

Parallel Processing for MPEG2 Encoding on OSCAR Chip Multiprocessor

Takeshi Kodaka[†], Hirohumi Nakano[†], Keiji Kimura[†] and Hironori Kasahara [†]

This paper proposes a coarse grain task parallel processing scheme for MPEG2 encoding using data localization which optimizes execution efficiency assigning coarse grain tasks accessing the same array data on the same processor consecutively on a chip multiprocessor and data transfer overlapping technique which minimize the data transfer overhead by overlapping task execution and data transfer. Performance of the proposed scheme is evaluated. As the evaluation result on an OSCAR chip multiprocessor architecture, the proposed scheme gave us 1.24 times speedup for 1 processor, 2.47 times speedup for 2 processors, 4.57 times speedup for 4 processors, 7.97 times speedup for 8 processors and 11.93 times speedup for 16 processors respectively against the sequential execution on a single processor without the proposed scheme.

1　はじめに

近年，ディジタル TV や DVD プレーヤー，ディジタルビデオカメラなどのディジタル情報機器では，マルチメディア処理が重要なプリケーションの一つとなっている．そのため，マルチメディアアプリケーションを効率的に処理できるデバイスの開発が求められている．これらのデバイスは操作の快適性や高品質の要求などから高パフォーマンスであることに加え，低コスト，低消費電力であることも望まれている．

従来のメディア処理用プロセッサアーキテクチャは，パイプライン化やスーパースカラ命令発行，SIMD 命令，VLIW などを用いて演算の高速化を行なってきた．ただし，これらの手法は主に命令レベルの並列性を用いているため命令レベルの並列性の限界により今後の性能向上には限界がある．そのため，よりスケーラブルな性能向上を得るためのアーキテクチャとしてチップマルチプロセッサが注目されている．チップマルチプロセッサアーキテクチャは，並列性の大きいループイタレーションレベルの並列性や，サブルーチンやループ間の粗粒度タスクレベルの並列性に加え，より小さな並列処理粒度のステートメントレベルの近細粒度並列性の利用が可能であり，さまざまな粒度の並列性を柔軟に利用したパフォーマンス向上が行なえる．しかし，チップマルチプロセッサアーキテクチャはスーパースカラアーキテクチャのようにハードウェアによる並列性の抽出などを行なっていないためチップマルチプロセッサ上で効率的な演算を行なうためにはチップマルチプロセッサに対応したソフトウェアによる最適化が必要である．

本論文では，動画像処理として広く用いられており，処理が重い MPEG2 エンコード処理に対する，チップマルチプロセッサ上でのメモリ利用最適化およびデータ転送最適化を伴う粗粒度タスク並列処理を提案し，OSCAR チップマルチプロセッサ上での性能について述べる．

以下，2 章で，対象とするアーキテクチャである OSCAR チップマルチプロセッサアーキテクチャの説明と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手法を提案し，3 章で OSCAR チップマルチプロセッサ上での提案手法の性能評価結果について述べ，4 章で結論を述べる．

† 早稲田大学理工学部コンピュータ・ネットワーク工学科
Dept. of Computer Science, Waseda University

2　MPEG2エンコードの並列処理

本章では，まず，OSCAR チップマルチプロセッサアーキテクチャについて説明し，参照した MPEG2 エンコードアルゴリズムについて述べた後，MPEG2 エンコードのチップマルチプロセッサ上での効率的な並列処理手法を提案する．

2.1　OSCARチップマルチプロセッサアーキテクチャ

本節では，本論文で対象とするチップマルチプロセッサアーキテクチャである OSCAR チップマルチプロセッサアーキテクチャ（OSCAR CMP）[1] およびそのプロセッサコアアーキテクチャについて述べる．

OSCAR CMP のネットワークおよびメモリアーキテクチャは，図 1 に示すように CPU，データ転送を CPU の処理とオーバーラップして行なえるデータ転送ユニット (DTU)，各々の CPU で実行するプログラムを格納するローカルプログラムメモリ (LPM)，PE 固有のデータを保持するローカルデータメモリ (LDM)，アドレスがグローバルアドレス空間にマップされており自 PE と他 PE の双方から同時にアクセス可能なマルチポートメモリの分散共有メモリ (DSM) を持つプロセッサエレメント (PE) を相互接続網 (バス結合，クロスバ結合など) で接続し 1 チップ上に搭載し，各 PE で共有するデータなどを格納

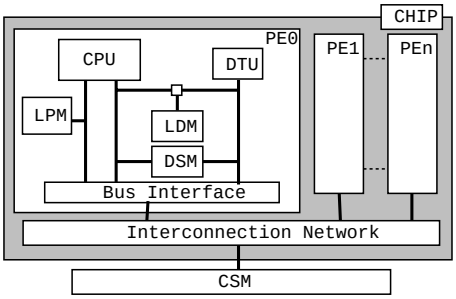

図 1: OSCAR CMP アーキテクチャ

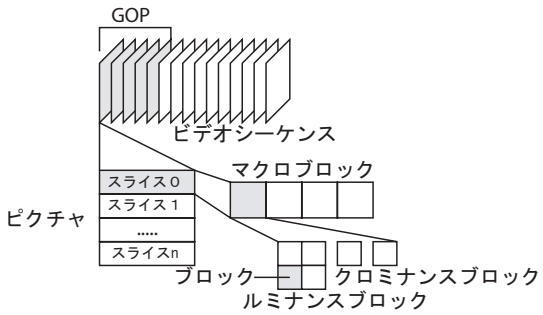

図 2: MPEG2 データ構造

する集中共有メモリ (CSM) を PE 外部に接続したアーキテクチャである．

2.2 MPEG2 エンコードアルゴリズム

本論文では，MPEG2 エンコードアルゴリズムの参照実装として，MediaBench[2] に収録されている MPEG2 エンコードプログラムである “mpeg2encode” を用いる．なお，以降説明するアルゴリズムは MediaBench でベンチマークパラメータで用いられているエンコーディングオプションでの動作を仮定する．

まず，MPEG2 ビデオのデータ構造の説明を行なう．MPEG2 ビデオのデータ構造は，図 2 に示すように階層的な構造をしている．まず，MPEG2 ビデオデータすべてのことであるビデオシーケンスがあり，内部データは，開始，終了を示すシーケンスヘッダと 1 つまたは複数のグループオブピクチャ(GOP) で構成される．GOP は，いくつかの画像フレームをグルーピングして構成される．ピクチャは画像フレーム一つ一つのことでありその内部はいくつかのスライスからなっている．スライスはピクチャの左上から始まり右下へとスキャン順に続く任意個のマクロブロックの集合で構成される．マクロブロックは，本論文でのエンコードパラメータである 4:2:0 カラーフォーマットでは，マクロブロックは，16×16 ピクセルブロックのルミナンスブロックと 2 つの 8×8 ピクセルブロックのクロミナンスブロックから構成される．最も小さいデータ単位が 8×8 ピクセルブロックのブロックである．

次に，MPEG2 エンコードの処理内容について説明する．MPEG2 エンコードは，大きく分けて動き推定，動き予測，DCT モード選択，データ変換，ビットストリーム出力，逆量子化，逆データ変換の 7 つのステージからなる．動き推定ステージは，動きベクトルの探索を行ないピクチャタイプにより符合化モードの設定を行なう．動き予測ステージは，動き推定で求めた動きベクトル，符合化モードに基づいて符合化対象ピクチャを生成する．DCT モード選択ステージは，符合化対象ピクチャに対してマクロブロックレベルでフレーム構造離散コサイン変換 (Discrete Cosine Transform:DCT) を適用するかフィールド構造 DCT を適用するかを決定する．データ変換ステージは，DCT 変換モードに基づき符合化対象ピクチャに DCT を適用する．ビットストリーム出力ステージは，DCT を適用したピクチャに対し量子化の適用を行ない，各種ヘッダの送出，ビットストリーム出力を行なう．また，ビットレート補償のためのビットレート制御および量子化係数の算出もこのステージで行なう．逆量子化ステージは，後続のエンコーディングで参照するピクチャを既にエンコードした画像から生成するために，量子化適用後のピクチャに対して逆量子化を適用する．逆データ変換ステージは，逆量子化適用後のピクチャに対して逆 DCT を適用し後続のエンコーディング参照するピクチャを生成し，画像バッファへの格納を行なう．なお，MPEG2 エンコードは主にマクロブロックレイヤーにおいて各マクロブロック単位に符合化が行なわれ，画像フレームの左上から右下へと順番にスキャンされエンコードされる．

2.3 並列性の抽出

まず，本論文で利用する並列性の抽出方法 [3] について述べる．はじめに，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)) に分割する．ここで，BPA は基本的には通常の基本ブロックであるが，並列性抽出のために単一の基本ブロックを複数に分割したり，逆に複数の基本ブロックを融合して一つの BPA を生成する．MT 生成後，コンパイラは BPA，RB，SB 等の MT 間の制御フローとデータ依存を解析し，結果をマクロフローグラフ (MFG)[3] として表す．次に，MFG から MT 間の並列性を最大限に抽出するためにデータ依存と制御依存を考慮し，各 MT が最も早い時点で実行可能となる条件解析である最早実行可能条件解析 [3] が行なわれる．その結果は，各 MT の制御依存とデータ依存を表したマクロタスクグラフ（MTG）[3] として表現される．MTG 生成後，MTG 上の MT をプロセッサあるいは複数のプロセッサエレメント (PE) をグループ化したプロセッサグループ (PG) に割り当てる．なお，このグループ化はプログラム中の各部分の並列性に応じソフトウェア的に行なわれる仮想的なものでハードウェア的なグループ化とは異なる．ここで，PG に割り当てられた繰返しブロックが，イタレーションレベルでデータ依存がなく並列処理可能な Doall ループの場合は，PG 内 PE 間でループ並列処理が行なわれる．

MPEG2 エンコードでは，マクロブロックを単位とした符合化が行なわれるため，マクロブロックレベルの並列性に注目し並列性抽出について考察する．マクロブロックレベルの並列性を抽出して利用する場合，ビットストリーム出力順やビットレート補償のための係数など直前のマクロブロックのエンコーディング情報を必要とするビットストリーム出力ステージ以外は，各符合化ステージにおいてそれぞれのマクロブロックは独立して演算できる．図 3(a) は，MediaBench での実装を基にした MPEG2 エンコーディングのコード例を示している．このコードから並列性の抽出を行なうと，各ステージがそれぞれ MT と

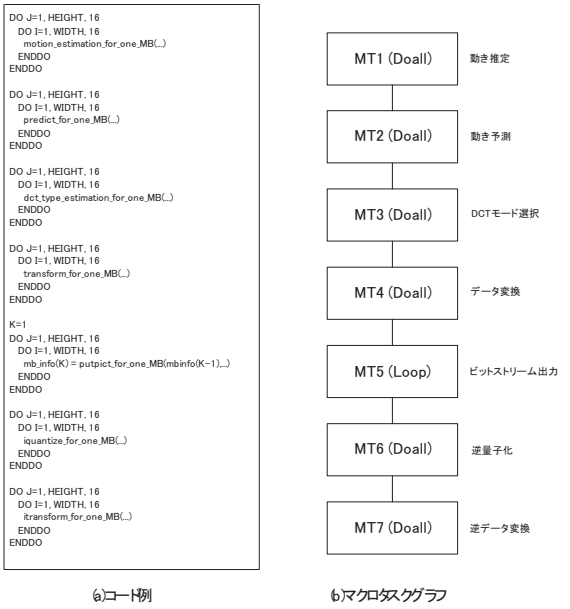

図 3: MPEG2 エンコードのコード例とマクロタスクグラフ

して図 3(b) のような MTG が生成される．MTG 中，各ノードは MT を表し，各エッジはデータ依存を表している．図 3(b) より，ビットストリーム出力ステージのみ，イタレーション間でループキャリー依存の存在により，シーケンシャルループであるが，他の各ステージからはマクロブロックレベルの並列性によりイタレーションレベルのループ並列性が抽出される．

2.4 データローカライゼーション手法

メモリウォール問題の深刻化により，CPU の処理速度とメモリアクセス速度の差が問題となってきており，CPU 近傍の高速なメモリの利用効率の向上が重要課題となっている．このメモリ利用最適化法の一つとしてループなどの粗粒度タスク間のデータローカリティ利用の向上をタスクの分割とタスク実行順序の並び替えによって実現するデータローカライゼーション手法 [4),5)] が提案されている．データローカライゼーション手法は，まず，データを共有する各ループ (MT) のデータ使用範囲が一致するように考慮しながら，そのデータ使用容量がプロセッサ内ローカルメモリ以下となるようにループ整合分割 [4)] を行なう．次に，分割したそれぞれの小ループを粗粒度タスクとして定義し，最早実行可能条件解析を適用し並列性を抽出しマクロタスクグラフを生成する．そして，マクロタスクグラフ上でマクロタスク間のデータ共有量を計算し共有量の多いマクロタスク群をデータローカライゼーショングループ（DLG）[4)] として定義する．その後，同一 DLG 内マクロタスクを同一プロセッサ上でなるべく連続して実行するように各タスクをプロセッサ上にスケジューリングする．以上により，データを共有する複数の粗粒度タスク間でプロセッサ内ローカルメモリを介した効率の良いデータの受渡しが可能となり，データローカリティを利用した，メモリ利用効率の最適化が可能となる．

2.3 節で抽出したループ並列性を用いた場合，チップ内メモリの利用に関して以下のような問題が発生する．MPEG2 のエンコード処理は一つのステージをピクチャ全体に対して適用した後に次のステージへと進む構造をしている．そのため，各ステージの処理開始時のイタレーションで演算されたエンコードデータは後続イタレーションを演算する際，チップ内ローカルメモリの容量はピクチャ全体のデータを保持できるほど大きくないためチップ内ローカルメモリに保持しておけない．そのため，高速アクセスが可能な CPU 近傍のチップ内ローカルメモリから，低速アクセスの大容量チップ外メモリへエンコードデータのストアが必要となる．また，後続のステージを演算する際にも，チップ外メモリにストアされている前ステージのエンコードデータをチップ内ローカルメモリへ転送する必要がある．そのため，チップ内ローカルメモリ–チップ外メモリ間のロード・ストアによる転送オーバーヘッドが発生し全体的な実行効率が低下する．

ここで，本論文で提案する MPEG2 エンコードにおけるデータローカライゼーション手法について説明する．まず，チップマルチプロセッサのチップ内メモリ容量を考慮し，マクロブロックレベルで各ステージを部分ループに整合分割する．このとき，ビットストリーム出力ステージは，シーケンシャルループではあるが，基本的にマクロブロックレベルで処理を行なうループとなっているため，他のステージと同様にマクロブロックレベルの部分ループに分割する．ループ整合分割後，分割された部分ループを粗粒度タスク（MT）として定義し並列性の抽出を行なう．MPEG2 エンコードでは各 MT は一つのマクロブロックのエンコーディング処理を行なう粒度となる．例えば，ピクチャの大きさが 4 つのマクロブロックである場合の並列性を抽出した結果は，図 4 のマクロタスクグラフ（MTG）として表される．図 4 では，各ノードは MT を表し，各エッジはデータ依存を表している．ノード中の $MTx\_i, (x = 1, \ldots, 7, i = 1, \ldots, N : N$ はピクチャ中の総マクロブロック数) は，$x = 1$ が動き推定ステージ，$x = 2$ が動き予測ステージ，$x = 3$ が DCT モード選択ステージ，$x = 4$ がデータ変換ステージ，$x = 5$ がビットストリーム出力ステージ，$x = 6$ が逆量子化ステージ，$x = 7$ が逆データ変更ステージをそれぞれ示し，i はエンコード対象とするマクロブロックのスキャン番号（ピクチャの右上から左下へスキャンされる）を示す．図 4 より，ビットストリーム出力ステージ以外ではマクロブロックレベルにおいてデータ依存が存在しないことがわかる．ここで，ビットストリーム出力ステージにおけるデータ依存エッジに関してエッジが接続されているマクロタスク間でのデータ共有量について考える．$MT5\_1$ から $MT6\_1$ および $MT5\_2$ へ接続しているデータ依存エッジを例とすると，$MT5\_1$ から $MT6\_1$ へのデータ依存エッジにおけるデータ共有量は $MT5\_1$ で量子化されたマクロブロックの演算データおよび量子化係数などである．対して $MT5\_1$ から $MT5\_2$ へのデータ依存エッジは，ループ整合分割を行なう前で発生していたループキャリー依存に関するデータ依存であり，ビットレート補償のための量子化係数，ビットレート係数やビットストリーム出力位置情報などである．これらのデータ共有量を比較すると

($MT5\_1$ から $MT6\_1$ 間でのデータ共有量)
$>$ ($MT5\_1$ から $MT5\_2$ 間でのデータ共有量)

となりデータ転送の効率化の観点から $MT5\_1$ 実行後，連

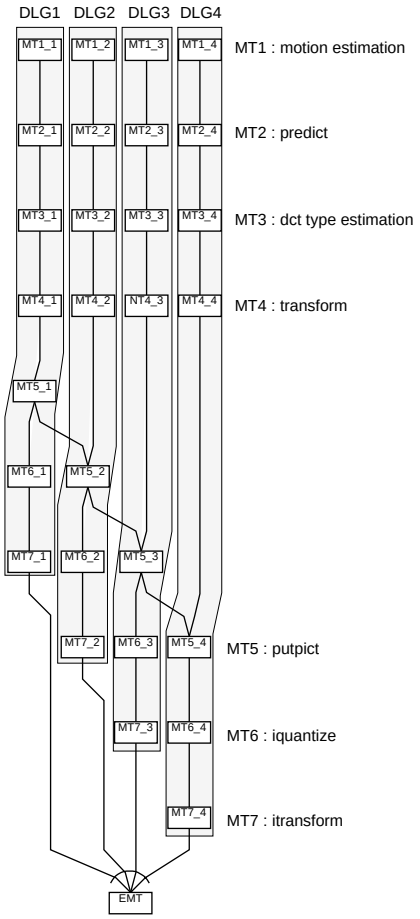

図 4: データローカライゼーション手法適用後の MPEG2 エンコードのタスクグラフ

続して $MT6\_1$ を実行しチップ内メモリを介したデータの授受を行なう方が効率的である．以上より，各 MT のプロセッサへのスケジューリングは，同じマクロブロックをエンコードする各ステージを 1 つのデータローカライゼーショングループ（DLG）として定義し，1 つのマクロブロックのエンコードは同一プロセッサで連続して行なうようにスケジュールする．例として，総マクロブロック数が 8 個の時のスケジュール結果を図 5 に示す．

2.5 データ転送オーバーラップを考慮したスケジューリング

データローカライゼーション手法により，データローカリティを最大限に活用してもローカルメモリへの初期データのロードや演算結果のストアなどのデータ転送が発生するため，データ転送オーバーヘッドによる速度低下が発生する．そこで，CPU と非同期にデータ転送を行なうデータ転送ユニット（DTU）を利用し，CPU でのタスク実行と DTU を利用したデータ転送をオーバーラップすることでデータ転送オーバーヘッドを隠蔽する．

ここで，ロードオーバーヘッド隠蔽技術は，あるタスク T_i の実行で必要なデータ D_i を外部メモリからプロセッサローカルメモリにロードするとき先行して実行されるタスクの実行中に DTU を利用して D_i をロードすることにより，本来は T_i の先頭で行なわれていた D_i のデータロードによるオーバーヘッドの隠蔽を行なうことを示し，これをプレロードと呼ぶ．また，ストアオーバーヘッド隠蔽技術は，あるタスク T_j において T_j の演算結果である D_j をプロセッサローカルメモリから外部メモリへストアする必要がある場合，T_j 終了時にストアはせず，後続に実行されるタスクの実行中でネットワークおよび共有メモリポートが空いているとき，DTU を利用し D_j をストアすることにより D_j のデータストアによるオーバーヘッドの隠蔽を行なうことを示し，ポストストアと呼ぶ．本節では，プログラム実行とデータ転送をオーバーラップすることでデータ転送オーバーヘッドを隠蔽するプレロード・ポストストア手法を考慮したスケジューリングの提案を行なう．

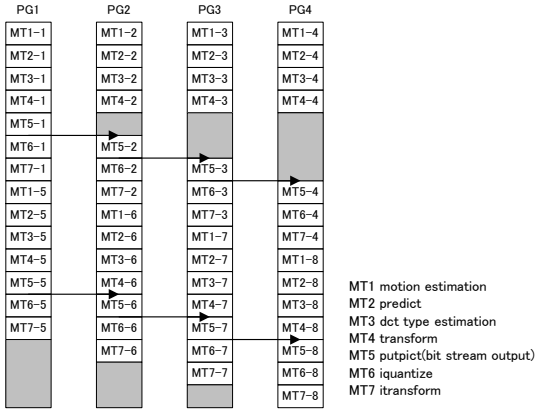

図 5: データローカライゼーションを適用したスケジューリング

ここで，前提条件として，プレロードおよびポストストアは MT の先頭または末尾においてのみ開始できるものとし，複数のプレロード，ポストストアの開始が登録できるものとする．ただし，一つの DTU では同時に一つのプレロード，ポストストアのみ実行できるものとし，登録順にプレロード，ポストストアを実行する．以下，プレロード，ポストストアを考慮したスケジューリング手法を提案する．

2.5.1 プレロード・ポストストア開始可能時刻の推定

まず，プレロードスケジューリングの初期開始時刻となる，プレロード開始可能時刻の設定を行なう．プレロード対象データは，あるマクロタスク MT_{pl} に生きて入り，かつ，MT_{pl} 内で前方露出参照されるデータであり MT_{pl} 実行開始時にプロセッサローカルメモリへデータ転送が必要なデータ D_{pl} で定義される．ここで，MT_{pl} はプロセッサ PG_l にスケジュールされたものとする．このとき，プレロード開始可能時刻を決めるために，まず，MT_{pl} 開始時刻から部分スケジュールを時間軸上前方向へスキャンし D_{pl} を最後に定義するマクロタスクを探索し，そのマクロタスクの末尾をプレロード開始可能時刻と設定する．このとき，D_{pl} のデータロードに必要なコストを推定し $Cost_{pl}$ とする．

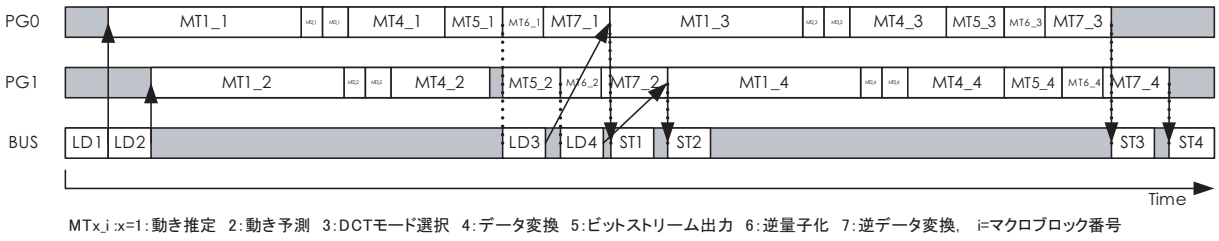

図 6: データ転送オーバラップを考慮したスケジューリング

次に，ポストストアスケジューリングの初期開始時刻となる，ポストストア開始可能時刻の設定を行なう．ポストストア対象データは，PG_m 上の MT_{ps} で定義されるデータ D_{ps} が後続の PG_n に割り当てられた MT'_{ps} で使用される場合の D_{ps} である．そのため，MT'_{ps} がスケジューリングされるまでポストストアが必要か判断できないため MT'_{ps} のスケジューリングが確定した時点でポストストア開始可能時刻を推定する．まず，MT'_{ps} のスケジューリングが決定したとき，MT_{ps} の終了時刻をポストストア開始可能時刻と設定する．このとき，D_{ps} のデータストアに必要なコストを推定し $Cost_{ps}$ とする

2.5.2 プレロード・ポストストアスケジューリング

スケジューリング対象となる MT のプレロード開始可能時刻またはポストストア開始可能時刻の設定ができたら，次に，プロセッサローカルメモリ–外部メモリ間のネットワーク利用状況，および，データ転送コストを考慮し，以下のようにプレロード・ポストストア開始時刻を決定する．プレロードおよびポストストアの開始時刻の評価は，プレロード，ポストストア共に同時に行なう．スケジューリングに必要なパラメータ以下のように再定義する．プレロード対象データ D_{pl} およびポストストア対象データ D_{ps} をプレロード・ポストストア対象データ D_{plps}，プレロード・ポストストア対象データ D_{plps} のロード・ストアに必要なコスト $Cost_{pl}$ および $Cost_{ps}$ を $Cost_{plps}$，プレロード対象データを利用するマクロタスクの MT_{pl} およびポストストア対象データを利用するマクロタスク MT'_{ps} を MT_{plps} とする．

1. プレロード･ポストストア開始可能時刻から $Cost_{plps}$ の時間分ネットワークが空いている場合，評価しているプレロード・ポストストア開始可能時刻をプレロード・ポストストア開始時刻として決定する．
2. 評価しているプレロード・ポストストアが開始可能時刻において，他のプレロード・ポストストアと重複し $Cost_{plps}$ の時間分ネットワークが空いていない場合は，重複しているプレロード・ポストストアと

   (MT_{plps} の開始時刻)
   – (プレロードポストストア開始可能時刻)

   の値を比較し，値が小さいプレロード・ポストストア開始可能時刻を優先して，1. に従いプレロード・ポストストア開始時刻を決定する．
3. 評価している時刻では，プレロード・ポストストア開始時刻が決定できない場合は，時間軸で直後の MT の終了時刻をプレロード・ポストストア開始可能時刻と再設定し，1. からプレロード・ポストストア開始時刻を決定する．

4 つのマクロブロックをエンコードする MPEG2 エンコードを例とし，提案したデータ転送オーバーラップを考慮したスケジューリングに従いプレロード・ポストストアを決定すると，図 6 のようなスケジューリング結果を得ることができる．図 6 は，1 バス構成のアーキテクチャ上で 2 プロセッサ用いた場合のスケジューリング状態を示し，時間軸は左から右へ時間が進行し，各スケジュールチャートは，上から PG0 の CPU 上でのタスクの実行状況，PG1 の CPU 上でのタスクの実行状況，バスの利用状況を示す．各 PG のスケジュールチャートにおいて，MTx_i はマクロタスクの実行を示す．バスの LD_j は j 番目のマクロブロックのエンコード実行に必要なデータのロードを実行中を示し，ST_k は k 番目のマクロブロックでの必要なストアを実行中であることを示す．また，グレーのチャートは，アイドル状態であることを示す．

3 性能評価

本章では，MPEG2 エンコードに対して 2 章で提案した並列処理手法を適用し OSCAR チップマルチプロセッサ (OSCAR CMP) 上で評価した結果について述べる．

3.1 評価条件

評価に用いるプログラムは，MediaBench に収録されている MPEG2 エンコードプログラム “mpeg2encode” を参照実装し，OSCAR 自動並列化コンパイラを利用するために Fortran で実装されたプログラムである．並列性の抽出は参照実装したシーケンシャルプログラムを OSCAR 自動並列化コンパイラを用いて並列性の抽出を行ない，手動でマクロタスクのスタティックスケジューリングを適用し，OSCAR CMP 用バイナリーコードを生成した．入力画像は，MediaBench で用いられる入力画像（compo.tar.gz 中の rec*.[YUV]）をシミュレーション時間短縮のために 256×256 ピクセルに縮小した画像を用い 4 フレームのエンコードを行ない，エンコードオプションは MediaBench で用いられているものと同一とする．

性能評価には OSCAR CMP をクロックレベルでシミュレートする詳細なシミュレータを用いた．OSCAR CMP のパラメータは，各メモリサイズ，アクセスレイテンシは，LDM の容量は 128K バイトとしアクセスレイテンシは 1 クロック，同様に DSM の容量は 128K バイトで自 PE 内のローカルアクセスには 1 クロック，他 PE へのリモートアクセスには 4 クロックかかるとし，CSM は本論文で使用する MPEG2 エンコードで利用するメモリ容量が十分確保されているものとしてアクセスレイテンシ

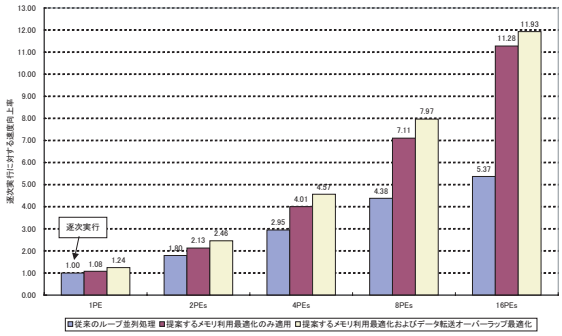

図 7: MPEG2 エンコード評価結果

は 24 クロックとした．各 PE が持つ CPU 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あり，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．

3.2 評価結果

OSCAR CMP 上での性能評価結果を図 7 に示す．図 7 中横軸の “1PE”，“2PEs”，“4PEs”，“8PEs” および “16PEs” は，使用プロセッサ数をそれぞれ示し，縦軸は逐次実行時間に対する速度向上率を示す．3 本の棒グラフのうち左側は，従来のマルチプロセッサシステム用並列化手法であるループ並列処理を適用した場合の性能，中央は提案手法のうちメモリ利用最適化技術であるデータローカライゼーション手法のみを適用した場合の性能，右側は本論文で提案したデータローカライゼーション手法およびデータ転送オーバーラップスケジューリングを適用した場合の性能をそれぞれ示す．

性能評価結果より，従来の並列化手法であるループ並列処理適用時は逐次実行に対して，2 プロセッサ利用時 1.80 倍，4 プロセッサ利用時 2.95 倍，8 プロセッサ利用時 4.38 倍，16 プロセッサ利用時 5.37 倍の速度向上率がそれぞれ得られた．提案手法のデータローカライゼーション手法のみを適用した場合，1 プロセッサ利用時 1.08 倍，2 プロセッサ利用時 2.13 倍，4 プロセッサ利用時 4.01 倍，8 プロセッサ利用時 7.11 倍，16 プロセッサ利用時 11.28 倍の速度向上がそれぞれ得られた．さらに，データローカライゼーション手法に加えデータ転送オーバーラップスケジューリングを適用した場合，1 プロセッサ利用時 1.24 倍，2 プロセッサ利用時 2.46 倍，4 プロセッサ利用時 4.57 倍，8 プロセッサ利用時 7.97 倍，16 プロセッサ利用時 11.93 倍の速度向上率が得られた．以上より，データローカライゼーション手法およびデータ転送オーバーラップスケジューリングを利用した提案手法は，従来手法であるループ並列化による並列処理に比べ，利用プロセッサ数が同一のときで，2 プロセッサ利用時 1.37 倍，4 プロセッサ利用時 1.55 倍，8 プロセッサ利用時 1.82 倍，16 プロセッサ利用時 2.22 倍の速度向上率が得られ，よりスケーラブルな性能を得ることが確かめられた．

4 まとめ

本論文では，複数ループにまたがるグローバルデータローカリティ最適化を行なうデータローカライゼーション手法と粗粒度タスクの実行とデータ転送をオーバーラップさせデータ転送オーバーヘッドを最小化するデータ転送オーバーラップスケジューリングから成るチップマルチプロセッサ上での MPEG2 エンコードの粗粒度タスク並列処理手法を提案し，その性能評価を行なった．その結果，OSCAR チップマルチプロセッサ上にて提案手法は，逐次実行に対し，1 プロセッサ利用時 1.24 倍，2 プロセッサ利用時 2.46 倍，4 プロセッサ利用時 4.57 倍，8 プロセッサ利用時 7.97 倍，16 プロセッサ利用時 11.93 倍の速度向上率が得られ，提案した MPEG2 エンコードの並列処理手法はチップマルチプロセッサ上で，スケーラブルかつ効果的な並列処理を実現できることが確認できた．

今後の課題として，MPEG2 エンコードのさらなる高速化として MPEG2 エンコードへのマクロブロック処理間の並列性とマクロブロック処理内での並列性という階層的な並列性を利用したマルチグレイン並列性の適用，及びキャッシュ共有型など他のチップマルチプロセッサアーキテクチャとの比較が挙げられる．

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」，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」，NEDO「先端ヘテロジニアスチップマルチプロセッサ研究開発事業」，文部科学省科学研究費補助金若手研究（B）（課題番号 15700074）及び特別研究員奨励費（課題番号 1501202）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本俊介氏（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），枝廣正人氏（NEC）に感謝致します．

参考文献

[1] 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*, Vol. E86-C, No. 4, pp. 570–579 (2003).

[2] C. Lee, M. Potkonjak and W. H. Mangione-Smith: MediaBench: A Tool for Evaluating and Synthesizing Multimedia and Communications Systems, *30th International Symposium on Microarchitecture (MICRO-30)* (1997).

[3] H. Kasahara, M. Obata and K. Ishizaka: Automatic Coarse Grain Task Parallel Processing on SMP using OpenMP, *Proc. 12th Workshop on Languages and Compilers for Parallel Computing* (2000).

[4] 吉田明正, 越塚健一, 岡本雅巳, 笠原博徳: 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 (1999).

[5] Ishizaka, K., Obata, M. and Kasahara, H.: Coarse Grain Task Parallel Processing with Cache Optimization on Shared Memory Multiprocessor, *Proc. of 14th International Workshop on Languages and Compilers for Parallel Computing (LCPC2001)* (2001).