Vol.2010-ARC-187 No.22
Vol.2010-EMB-15 No.22
2010/1/29

H.264/AVC エンコーダのマルチコアプロセッサにおける階層的並列処理

見神広紀[†]　宮本孝道[†]　木村啓二[†]　笠原博徳[†]

本稿ではビデオコーデックである H.264/AVC エンコーダの高速化手法としてフレームおよびマクロブロックでの階層的な並列処理を提案する．H.264/AVC エンコーダの一実装である x264 上にマクロブロックでの並列処理機能を実装し，64 コアのマルチコアシステム上での処理性能の評価を行った．その結果，2 コア集積のマルチコアである Intel Itanium2 (Montvale)を 32 基搭載した 64 コア構成の ccNUMA サーバである SGI Altix450 において，フレームでの並列処理のみの場合が 6.3 倍であったのに対しフレームおよびマクロブロックの 2 階層で行った場合は 10.6 倍の性能向上が得られた．

Hierarchical Parallel Processing of H.264/AVC Encoder on an Multicore Processor

Hiroki Mikami[†]　Takamichi Miyamoto[†]　Keiji Kimura[†]
and Hironori Kasahara[†]

This paper proposes hierarchical parallel processing method of H.264/AVC encoder. Data structures and data dependencies are analyzed to exploit multi-level parallelization as frame-level and macroblock-level. We implemented macroblock-level parallel processing on the x264, an open source H.264/AVC encoder. As a result, on SGI Altix450 (Intel Itanium2 (Montvale), 64 cores ccNUMA server), speed up is saturated by using 8 cores when execute encoder in only frame-level parallelization. However, scalable speedup is attained when execute encoder in frame and macroblock multi-level parallelization.

1. はじめに

HDTV やゲーム機，インターネットでの動画サービスの普及により大容量の動画を効率よく圧縮する技術の重要性が高まっている．

動画圧縮は人間の感覚から冗長と考えられる情報の削減，および空間的・時間的な共通部分の省略によって行う．動画圧縮方式であるH.264/AVC[1]は空間的・時間的な冗長性を取り除く多くの手法を適用し，従来の圧縮方式を上回る高い圧縮効率を達成している．

その一方で，H.264/AVC は非常に高い計算能力を要求するという問題がある．特にリアルタイム処理を行う場合，このような性質は大きな障害となる．動画圧縮を高品質で行うには，マルチコアプロセッサや特定用途向けのデバイスなどの計算能力の高いプラットフォームが必要である．

現在，高速化には DSP や専用デバイスなどのハードウェアによる手法が主に用いられている．しかしこれらのハードウェアの開発には高度な技術が必要であり，パラメータ設定に柔軟性に欠ける．さらに，サポートが求められる動画圧縮フォーマットの数も増加する一途であることを考慮すると，ソフトウェアによる手法を用い汎用プロセッサ上で並列処理を行うことで生産性及び機器構成自由度向上に寄与できると考えられる．

H.264/AVCエンコーダのソフトウェアによる並列性に関しては，Yen-Kuang ChenらはHyper-threadingにおいてスライスレイヤーとフレームレイヤーを組み合わせた並列処理を提案している[2]が，この手法では遅延が大きくデータがストリームで流れてくる場合やリアルタイム処理には不向きである．さらに，スライスレイヤーの並列性は符号化効率とのトレードオフであるため複数階層を用いても大きな並列性を引き出しにくい．またTom R. Jacobsらはスライスレイヤーでの並列処理を提案している[3]が，それと同時にビットレートが 20%程度増加するといった符号化効率の低下を指摘している．

本稿では，動画圧縮という観点から並列処理時の符号化効率の維持を重視し，並列処理による符号化効率の低下が理論上ゼロであるフレームレイヤーとマクロブロックレイヤーに注目した．さらに今後幅広く利用可能になると予測されるメニーコアでの適用を目指し，できるだけ多くの並列性を抽出したい．従って本稿ではこの 2 レイヤーを組み合わせて並列処理を行う“H.264/AVC エンコーダのフレームレイヤーとマクロブロックレイヤーでの階層的並列処理”を提案する．本手法により，符号化効率を維持したままコア数の増加とともに性能向上が可能であることを確認した．

[†] 早稲田大学 基幹理工学部 情報理工学科
Dept. of Computer Science and Engineering, Waseda University

Vol.2010-ARC-187 No.22
Vol.2010-EMB-15 No.22
2010/1/29

2. H.264/AVC エンコーダの並列性

本章では，まず H.264/AVC エンコーダのデータ構造およびアルゴリズムについて述べる．次に，よりよい高速化を行うための並列性の抽出手法および今回採用した手法の特徴について述べる．

2.1 H.264/AVC エンコードアルゴリズム

H.264/AVC ビデオのデータ構造 は図 1 に示すように階層的な構造をしている．

ビデオシーケンスはビデオデータ全体である．ビデオシーケンスはシーケンスヘッダとグループオブピクチャ（GOP）で構成される．GOP はいくつかのピクチャをグルーピングしてしたものである．ピクチャは画像フレーム 1 枚を指し，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 1 ピクチャのうち任意個のマクロブロックの集合であり，H.264/AVC ではスライスがエンコードの基本単位となる．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ており，予測の単位として用いられる．最後に最も小さいのが 8×8 および 4×4 ピクセルブロックのブロックレイヤーであり，DCT 処理の単位として用いられる．

次に，H.264/AVC エンコードの処理内容について説明する．H.264/AVC エンコードは，図 2 に示すように大きく分けて以下の 6 つのステージからなる．

動きベクトル検出

隣接するブロックまたは任意の 2 枚のピクチャから動きベクトルの探索を行う．

動き補償

ブロックレベルで適切なピクセルサイズを選択しつつ，すでに符号化したフレームから動きを予測する．

DCT/量子化

符号化対象ピクチャに対してブロックレベルで離散コサイン変換を適用し，係数行列の量子化を行う．H.264/AVC では 16bit 整数の精度で変換を行う．

エントロピー符号化

コンテキスト適応型可変長符号化（CAVLC）またはコンテキスト適用型算術符号化（CABAC）を行う．

逆量子化/逆 DCT

後続のエンコーディングで参照するピクチャを生成するために，量子化適用後のピクチャに対して逆量子化および逆変換を適用する．

デブロッキング・フィルタ

動き補償予測による予測誤差からブロックノイズの影響を除くために，ブロック境界の平滑化を行い，バッファに格納する．

H.264/AVC エンコードはこれらの処理がスライスレイヤーにおいて行われる．各マクロブロック単位で画像フレームの左上から右下へと順番に予測を行い，スライス単位で符号化される．

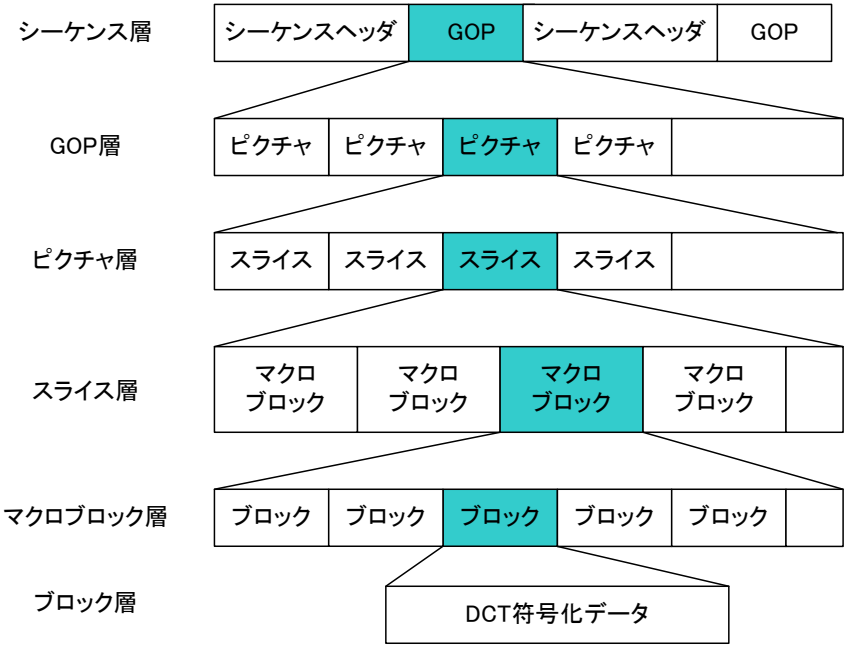

図 1 H.264/AVC ビデオのデータ構造

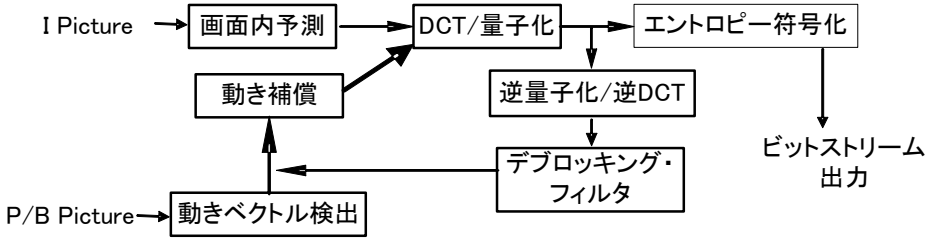

図 2 H.264/AVC エンコーダの構成（概略）

Vol.2010-ARC-187 No.22
Vol.2010-EMB-15 No.22
2010/1/29

2.2 抽出可能な並列性

H.264/AVC エンコードから並列性を抽出するために，データ構造に注目する．H.264/AVC ビデオデータは前節で述べたように階層的な構造をしており，各レイヤーのデータはランダムアクセス性の確保やエラー耐性の補償などのためにデータ同士でその独立性が確保されている部分がある．ここでは，各レイヤーでのデータの独立性に注目し，並列性の抽出を行なう．

GOP レイヤーでは, GOP 単位でのランダムアクセス性の確保のために各 GOP 間にはデータ依存がない.このため GOP 単位で効率のよい並列処理を行える．しかし GOP は複数のピクチャで構成されるため並列処理に伴う遅延が大きく，GOP の切れ目を符号化前に判断する必要があるため符号化効率も低下する．

ピクチャレイヤーでは，時間冗長度削減によるデータ圧縮のために各ピクチャは参照関係を持つことによるデータ依存関係がある．この時間冗長度削減方法は 3 つのタイプが定義されており，それぞれ I ピクチャ, P ピクチャ, B ピクチャと呼ばれる. I ピクチャは，他のピクチャの情報を使用せず自身のピクチャの情報のみで符号化を行なう．P ピクチャは，1 枚の I ピクチャまたは P ピクチャを参照ピクチャとし，動き予測で符号化される．B ピクチャは 2 枚の I ピクチャまたは P ピクチャを参照ピクチャとして予測符号化される. I/P フレームは他のフレームから参照されるが，B フレームは基本的に参照されないため，図 3 で示すように P フレームのエンコードが完了すれば後続の P/B フレームを並列処理することができる．本手法ではフレームを複数枚プールしておく必要があるため並列処理による遅延が生ずる．しかし，符号化効率は変化しない．

スライスレイヤーでは，エラー補償のために各スライス間は独立して符号化される．このためスライス単位で効率のよい並列処理を行える．さらに並列処理による遅延もゼロである．しかし空間冗長度削減のための探索空間が狭くなるため，スライス数を増やすに従って符号化効率が大幅に低下する．

マクロブロックレイヤーでは，図 4 のように左上・上・左の隣接マクロブロックの情報を元に動きベクトルの計算を行なうことによるデータ依存がある．この依存関係からマクロブロック単位で並列処理を行なうためには図 5 のようなwave-front手法を用いる必要がある[4]. 図 5 においてマクロブロック 1 行を同一プロセッサで処理する．従って並列処理効率はよくないが，並列処理に伴う遅延や符号化効率の低下は生じない．

以上のように，H.264/AVC エンコーダを並列処理する際にはトレードオフが存在する. 並列処理性能を重視する場合は, GOP またはスライスのレイヤーで行うとよいが，それにより符号化効率の低下が生じる．遅延を最小に押さえたい場合は，スライスやマクロブロックのレイヤーでおこなうのがよいが，前者には符号化効率低下，後者には並列処理性能があがらないといった問題がある．符号化効率を重視する際はフレームやマクロブロックで行うとよいが，並列処理性能が制限される．

本稿では、符号化効率を維持しつつ並列性を最大限利用するという目的で、フレームレイヤーとマクロブロックレイヤーの階層的な並列処理を行う。

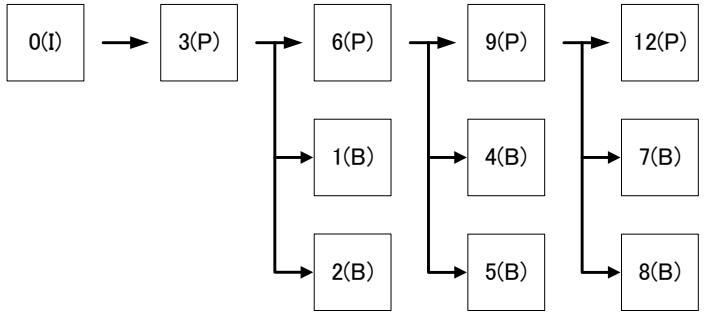

図 3　フレームレイヤーでの並列処理イメージ

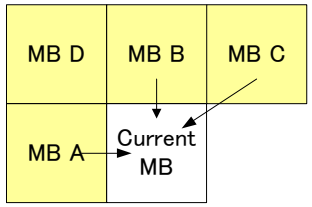

図 4　近接マクロブロックの参照関係

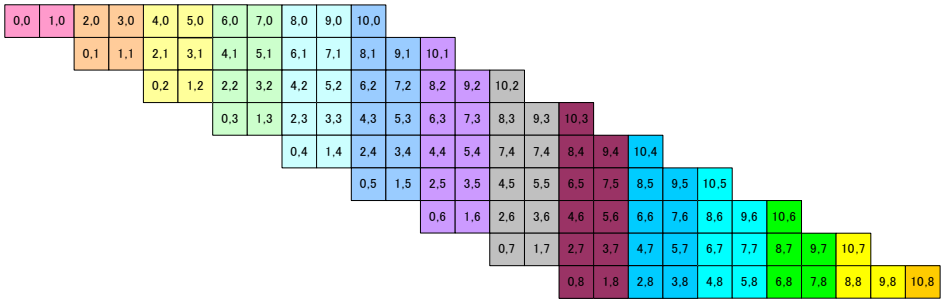

図 5　wave-front 手法を用いたマクロブロックレイヤーでの並列処理イメージ

Vol.2010-ARC-187 No.22
Vol.2010-EMB-15 No.22
2010/1/29

2.3 複数階層での並列処理の問題点

複数階層で並列処理を行う際は，限られたリソースを適切に割り振ることが重要である．本節ではマクロブロックレイヤーでの並列処理に注目し，プロセッサの割り当てについて述べる．

マクロブロックレイヤーでは wave-front 手法を用いて並列処理を行うために，以下のような問題点がある．

(1) **同時実行可能なマクロブロック数に限界がある**

(2) **プロセッサ間の同期が多くデータ転送も伴うためオーバーヘッドが大きい**

このため，潤沢なリソースがある環境でも単純に全てのプロセッサに処理を割り当てるのではなく適切な数だけ割り当てないと，効率のよい並列処理は実現できない．そこで，まず理論値による予測を行う．

1 ピクチャにおける横のマクロブロック数を w_{mb}，縦のマクロブロック数を h_{mb} とすると，同期のコストが限りなく小さい場合でもマクロブロックレイヤーで必要なプロセッサ数は最大で

$$\min\left(ceil\left(\frac{w_{mb}}{2} \right), h_{mb} \right)$$

である．図 6 に $w_{mb}=6$， $h_{mb}=8$ の場合を 4 プロセッサで処理した例を挙げる．図 6 が示すように，必要以上のプロセッサを割り当ててもプロセッサのアイドル時間が増えるだけでありエンコード速度の高速化に寄与しない．

また，理想的な場合においてマクロブロックレイヤーでの並列処理性能の最大は

$$\frac{w_{mb}+2(h_{wb}-1)}{w_{mb}h_{mb}}$$

となる. 図 7 に理想的な環境下で SD (640×480)の動画をマクロブロックレイヤーで並列化したときの並列処理性能を示す．プロセッサ数が少ない場合は効率的に並列処理されているのに対し，プロセッサ数が横のマクロブロック数の半分を超えた付近から急激に処理効率が落ちていることがわかる．

以上より，マクロブロックに割り当てるプロセッサ数を少なくし，残りのプロセッサをより粒度の高いレイヤーで並列処理することが良いとわかる．

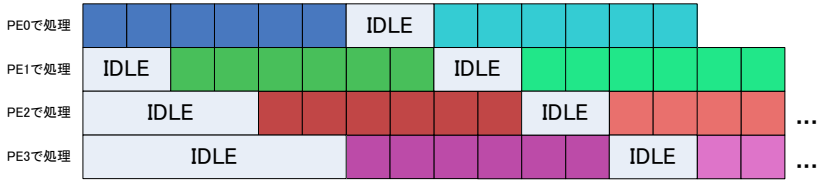

図 6　プロセッサが過剰な場合のマクロブロックレイヤー並列処理例

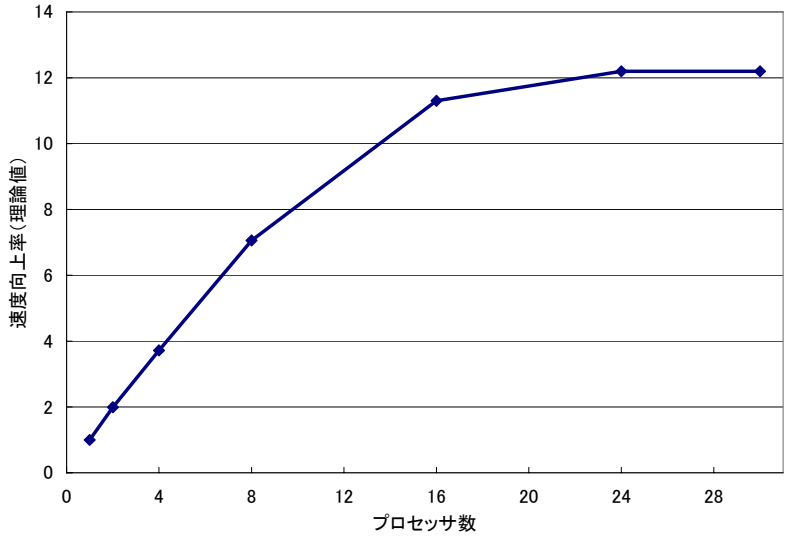

図 7　理想的環境下での SD 動画のマクロブロック並列処理性能　（理論値）

3. 性能評価

3.1 評価環境

評価アプリケーションとして，オープンソースで開発が進められているH.264/AVCエンコーダである“x264”[5]を使用した。x264 はすでにフレームレイヤーでの並列処理が実装されているためフレームレイヤーに関しては実装をそのまま使用し，マク

Vol.2010-ARC-187 No.22
Vol.2010-EMB-15 No.22
2010/1/29

ロブロックレイヤーでの並列処理を実装した．入力データとしては，PARSECベンチマーク[6]でも使用されているSD （640×320） 解像度のElephants Dream[7] 128 フレーム分を使用した．エンコードパラメータはベースラインプロファイルの品質固定圧縮モードで行なった．

また評価には 2 コア集積の Intel Itanium2（Montvale）を 32 基搭載したマルチコアシステムである SGI Altix450 を用いた．SGI Altix450 は 2 基の Intel Itanium2 とローカルメモリ（DDR2 DIMM）を Hub で接続したものを 1 ブレードとし，ブレード間を NUMALink で結合している．各プロセッサはローカルメモリを持つが，全メモリで論理的に共有メモリを構成する ccNUMA アーキテクチャのマルチコアシステムである．使用した Intel Itanium2 のメモリ構成は表 1 に示した．

表 1 Intel Itanium2（Montvale）のメモリ構成

CPU	Intel Dual-core Itanium2（Montvale） 1.66GHz
L1 I-cache	16KB/1PE
L1 D-cache	16KB/1PE
L2 D-cache	256KB/1PE
L2 I-cache	1MB/1PE
L3 cache	12MB/2PE

3.2 マクロブロックでの並列処理

図 8 にマクロブロックだけで並列処理した時の性能を示す．横軸はマクロブロックに割り当てたスレッド数，縦軸は速度向上率を示している．2 スレッドで処理した場合は 1.3 倍となったが，4 スレッドで処理した場合は逆に 0.7 倍となった．

これは使用した Intel Itanium2 が 2 コア集積であり，2 プロセッサまでは共有 L3 キャッシュを用いて同期およびデータ転送が行えるのに対し，4 プロセッサでは Hub を介した通信が必要である．Hub を介したメインメモリへのアクセスレイテンシは L3 キャッシュでの通信に比べ 40 倍遅く，オーバーヘッドが無視できなくなったものと考えられる．

3.3 複数階層での並列処理

図 9 にフレームレイヤーとマクロブロックレイヤーの複数階層で並列処理した時の，並列処理性能を示す．横軸は総スレッド数，縦軸は速度向上率を示している．右側の網掛けはフレームのみでエンコードを行った場合，左側の斜線はマクロブロックに 2 プロセッサを割り当てた場合の速度向上率を示している．SD 画像においてフレームレベルの並列処理のみで処理時間を比較したところ，逐次実行に対して 2 プロセッサ時で 1.9 倍，4 プロセッサ時で 3.6 倍，8 プロセッサ時で 6.3 倍の速度向上率が得られた.しかしながらフレームレベルのみでは 8PE で並列性能の限界に達している．
一方,フレームレベルとマクロブロックレベルで複数階層並列処理を行い処理時間 を比較したところ，2 プロセッサ使用時には 1.2 倍，4 プロセッサ時で 2.3 倍とフレームレベルのみでの並列性能に劣るものの，8 プロセッサ時で 4.6 倍，16 プロセッサ時で 7.9 倍，32 プロセッサ時で 8.7 倍，64 プロセッサ時で 10.6 倍と 8 プロセッサ以上でもコア数の増加とともに性能向上が得られた．

特に複数階層で並列化した場合，フレームに 8 プロセッサ以上を割り当てても速度向上が見られる. これは x264 のフレームでの並列処理の実装として 1 ピクチャが完全に符号化されるのを待たず，符号化可能な部分から随時実行を開始するため，1 ピクチャの処理時間が短くなると符号化を開始できるフレームが増えるためだと考えられる．

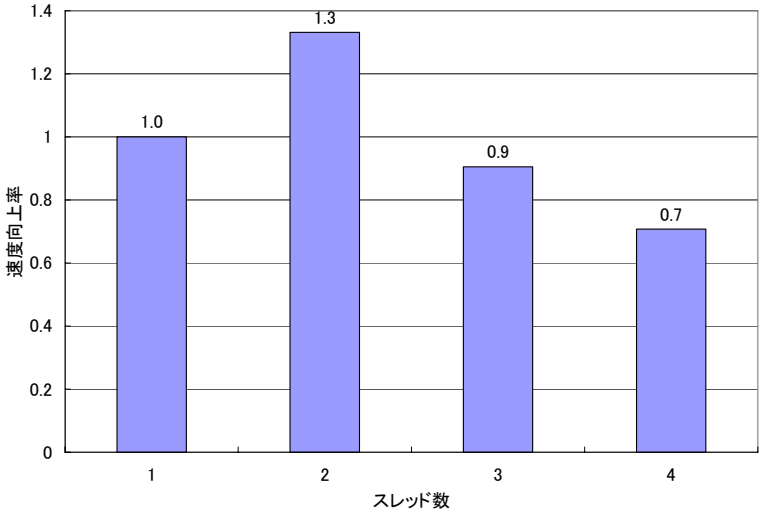

図 8 マクロブロックでの並列処理性能

Vol.2010-ARC-187 No.22
Vol.2010-EMB-15 No.22
2010/1/29

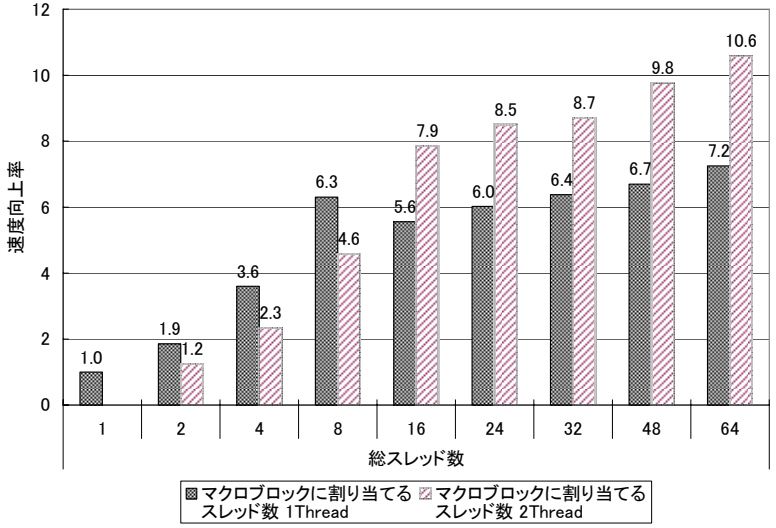

図 9 複数階層での並列処理性能

4. おわりに

本稿ではビデオコーデックである H.264/AVC エンコーダの高速化手法としてフレームおよびマクロブロックでの階層的な並列処理を提案した．オープンソース H.264/AVC エンコーダの x264 に対してマクロブロックでの並列処理を実装し，64 コアのマルチコアシステム上での処理性能の評価を行った．その結果，2 コア集積のマルチコアである Intel Itanium2（Montvale）を 32 基搭載した 64 コア構成の ccNUMA サーバである SGI Altix450 において，フレームでの並列処理のみの場合が 6.3 倍であったのに対しフレームおよびマクロブロックの 2 階層において適切にプロセッサを割り当てた場合は 10.6 倍の性能向上が得られ，フレーム並列処理の性能を引き出すことで 64 コアまでスケールアップすることが確認できた．

謝辞 本研究の一部は NEDO“情報家電用ヘテロジニアス・マルチコア技術の研究開発”，NEDO“低消費電力メニーコアプロセッサシステム技術の先導研究”および早稲田大学グローバル COE“アンビエント SoC”の支援により行われた．

参考文献

1) ISO/IEC 14496-10:2003. Coding of audiovisual objects - part 10: Advanced video coding (2003).
2) Yen-Kuang Chen, Xinmin Tian, Steven Ge, and Millnd Girkar. Towards efficient multi-level threading of h.264 encoder on intel hyper-threading architectures. Proc. of 18th International Parallel and Distributed Processing Symposium (2004).
3) Tom R. Jacobs, Vassilios A. Chouliaras, and David J. Mulvancy. Thread-parallel mpeg-2, mpeg-4 and h.264 video encoders for soc multi-processor architectures. IEEE Transactions on Consumer Electronics, 52(1):269–275, Feb (2006).
4) Z.Zhao and P.Liang. A highly efficient parallel algorithm for h.264 video encoder circuits and systems. Proc. of IEEE International Symposium on Circuits and Systems (2006).
5) x264 - a free h264/avc encoder. http://www.videolan.org/developers/x264.html
6) Christian Bienia, Sanjeev Kumar, Jaswinder Pal Singh and Kai Li. “The PARSEC Benchmark Suite: Characterization and Architectural Implications.” Technical Report TR-811- 08, Princeton University, Jan (2008).
7) Blender Foundation and Netherlands Media Art Institute. Elephants dream. http://www.elephantsdream.org/.