ハイパフォーマンス
コンピューティング 69－5
（1997. 12. 12）

# ベクトル計算機上での MPEG-2 ソフトウェアエンコーダの高速化について

杉原 光太、池川 将夫、土肥 俊

NEC C&C メディア研究所

MPEG-2 ソフトウェアエンコーダをベクトル計算機上で実現することを検討し、NEC 製スパコン SX-4 に実装 / 評価した結果について報告する。MPEG-2 エンコーダの核となる離散コサイン変換と動きベクトル探索について、そのベクトル計算機向きの高速化手法と性能評価について述べる。双方ともオリジナルのプログラムに比べてそれぞれ約 18 倍と約 6 倍の高速化を実現し、その結果、MPEG-2 エンコーダ全体は約 5 倍の高速化を実現した。今回開発したベクトル計算機向き MPEG-2 ソフトウェアエンコーダを、筆者らが開発している「実時間可視化システム RVSLIB」の動画生成ツールに組み込む予定である。これにより、動画化された大規模数値計算結果を MPEG-2 方式で圧縮し、手軽に通信や蓄積することを目指している。

# ON THE ACCELERATION OF AN MPEG-2 SOFTWARE ENCODER ON VECTOR COMPUTERS

Kouta Sugihara, Masao Ikekawa, and Shun Doi

C&C Media Research Laboratories, NEC Corporation

4-1-1, Miyazaki, Miyamae-ku, Kawasaki-shi, Kanagawa, 216 Japan

This paper considers the vectorization of the MPEG-2 software encoder. New vectorization techniques are introduced for the Discrete Cosine Transform (DCT) and the Motion Estimation (ME) calculations, which dominate computational costs in the encoder. Speedup obtained by these techniques are 18 and 6, respectively for the DCT and ME on the NEC SX-4 supercomputer. The overall speedup due to the vectorized MPEG-2 encoder ranges from 2.0 to 5.35, depending on the range of the motion vector search. The present techniques will be applied to the RVSLIB (Real-time Visual Simulation Library), which enables direct generation of an MPEG-2 movie file from on-going large-scale simulation on the SX-4 supercomputer.

## 1 はじめに

計算機の高速化、ネットワーク分散環境の普及により、高速な計算サーバ上での計算結果をユーザのクライアント上にオンラインで可視化表示したいという要求が高まっている。これらの要求を背景に当研究グループでは、従来よりスーパーコンピュータ (スパコン) 上でのシミュレーションの計算結果を、計算と同時に可視化・動画化する「リアルタイム可視化システム RVSLIB」 [1] を開発している。 RVSLIB では解析計算からレンダリング処理までを計算サーバ上で行い、ピクセル表示のみをクライアントで行う。このとき、サーバ側で作られたイメージデータをクライアントに転送する必要があり、その際のネットワークの負荷を軽減するため、動画像圧縮技術が必要になる。また計算終了と同時に、生成した計算結果のディジダル動画情報をファイルとして保存するためにも動画像圧縮技術は有用である。

本研究では動画像圧縮として MPEG-2[2] を取り上げ、ソフトウェアによるベクトル計算機上での高速 MPEG-2 エンコーダの開発について検討する。本報告では、 MPEG-2 エンコーダの処理の核となる離散コサイン変換・逆離散コサイン変換と動きベクトルの探索法について、ベクトル計算機向きアルゴリズムを開発し、それらを用いたエンコーダのベクトル計算機 SX-4 上での性能評価を行う。

離散コサイン変換・逆離散コサイン変換は通常、高速算法 [3] が用いられるが、ベクトル長が 8 と極めて短く、ベクトル計算機上では性能が期待できない。そこでベクトル長がピクチャ内の全ブロック数になるように変換し、高速化する。動きベクトルの探索は通常のマクロブロック間の距離計算のベクトル長が 16 と短く、ベクトル計算機では高速に実行されない。そこで探索範囲内の画素値の格納方式を変え、ベクトル長が横方向の探索範囲と縦方向の探索範囲の積になるようにし、高速化を図った。

本論文の第 2 章では MPEG-2 ソフトウェアエンコーダの概要について述べる。第 3 章、 4 章それぞれで開発した離散コサイン変換・逆離散コサイン変換と動きベクトルの探索法のベクトル計算機向きアルゴリズムについて詳述し、 SX-4 上での性能について報告する。第 5 章では開発したベクトル計算機向きの MPEG-2 エンコーダ全体の SX-4 上での性能について報告する。

## 2 MPEG-2 エンコーダ

MPEG とは、'Moving Picture Experts Group' の頭文字であり、その目的は動画像とオーディオの符号方式の標準化である。 MPEG-2 は現行放送用以上の品質で通信、放送、蓄積などに利用するための動画像圧縮方式である。 MPEG-2 エンコーダの構成を図 1 に示す。図 1 で示されるように処理の中心は動き推定の際の動きベクトルの検出と離散コサイン変換 (Discrete Cosine Transform : DCT)、逆離散コサイン変換 (Inverse Discrete Cosine Transform : IDCT)、量子化、逆量子化、符号化である。そこで MPEG-2 ソフトウェアエンコーダのベクトル計算機上での高速化にあたって、 DCT/IDCT と動きベクトルの検出処理のベクトル化による高速化を中心に行った。また本研究では MPEG Software Simulation Group によって作られたフリーウェアの mpeg2encode を利用した。

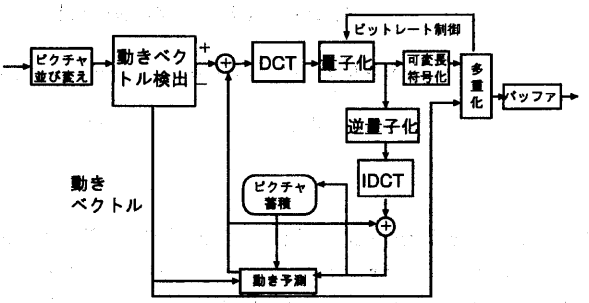



図 1:MPEG-2 エンコーダの処理構成

## 3 DCT/IDCTの高速化

本章ではDCT/IDCTについて述べ、その後開発したDCT/IDCTの高速化アルゴリズムについて説明し、そのSX-4上での評価結果について報告する。

DCTは以下のように定義される。

$$S_{vu} = \frac{1}{4} C_u C_v \sum_{x=0}^{7} \sum_{y=0}^{7} s_{yx} \cos \frac{(2x+1)u\pi}{16} \cos \frac{(2y+1)v\pi}{16}$$

$$C_u, C_v = \begin{cases} \frac{1}{\sqrt{2}} & \text{for } u, v = 0 \\ 1 & \text{otherwise} \end{cases}$$

通常のハードウェア、ソフトウェアでは乗算の回数を減らすことが処理の高速化には重要で、DCT/IDCTはいずれも乗算回数が少ないことを特徴とした高速算法(例えば[3])が用いられている。しかしそのような高速算法はアルゴリズムが複雑になるためベクトル長が短く、ベクトル計算機上では高速に実行されず、むしろ定義通りの変換式の8×8の行列演算の方が高速に実行される。しかし定義通りの変換式ではリストを使っても最大ベクトル長は64しかとれず、ベクトル計算機上で十分な性能は得られない。

DCT/IDCTはピクチャ内の8×8の全ブロックに対して行われるので、ここではピクチャ内のブロックに関するループとブロック内の8×8の画素領域に関するループを入れ換え、ピクチャ内の全ブロック数をベクトル長とすることで高速化を図った。但しこの演算では画素データのアクセスが64飛びになり、バンク競合が生じ、性能が低下する。そこでデータの格納形式を工夫することでブロックのサイズを疑似的に65にし、バンク競合による性能低下を避け、ベクトル化による効果を十分引き出せるようにした。

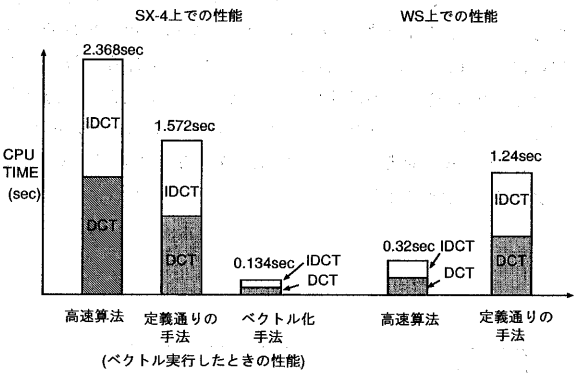



図2: DCT/IDCTに関する各算法の性能比較

DCT/IDCTに関し、高速算法を用いた場合と通常の定義式通りの8×8の行列演算を使う場合と今回試作したベクトル計算機向きの演算方法についてSX-4とワークステーションEWS4800(200MHz R10000プロセッサ)上で性能評価した結果を図2に示す。本稿を通じて数値実験には横720× 縦480 画素のflower gardenと呼ばれる画像の先頭3フレーム(ピクチャタイプ: I、P、P)を使用した。以後ワークステーションEWS4800(200MHz R10000プロセッサ)をWSと略す。

DCT/IDCTの実行状況の解析(図2)

- ベクトル計算機上では、高速算法に比べ今回開発したベクトル化手法の速度向上率は17.7となった。定義通りの手法と比較しても、今回開発したベクトル化手法は11.7倍の高速化を実現した。これは高速算法のベクトル長が8であるのに対し、今回開発したベクトル化手法のベクトル長がピクチャ内のブロック数(8100)であるためである。
- WS上での高速算法に比べ、SX-4上での今回開発したベクトル化手法の速度向上率は2.39となった。

## 4 動きベクトル探索の高速化

MPEG-2エンコーダにおける重要な技術として動き補償予測方式がある。これは動領域の検出を効果的に行うために提案された手法で、着目ピクチャと参照ピクチャ間で対象領域の動きベクトルを検出し、参照ピクチャにおいて動きベクトル分だけずらした位置を参照画素とし、これを予測値として着目画素との差分(予測誤差)を伝送する方

法である。MPEG-2エンコーダの処理の中では、この動きベクトル探索が最も大きなウェートを占める。以下に動きベクトルの検出法を説明する(図3)。

現在のピクチャ(m番目のピクチャとする)内の着目しているマクロブロック($16 \times 16$画素からなる領域)の左上端の画素位置を$(k,l)$とする。また、その画素値を$X_m(k,l)$とする。同様に$m-1$番目のピクチャ内の(k,l)での画素値を$X_{m-1}(k,l)$とする。このとき、

$$S_{ij} = \sum_{p,q=0}^{15} | X_m(k+p, l+q) - X_{m-1}(k+p+i, l+q+j) |$$

を最小とするようなベクトル$(i,j)(-L \leq i,j \leq L)$を見つけ、これを動きベクトルとする。ここでLを探索範囲と呼ぶ。

距離計算のときのベクトル長が短いため、動きベクトルの検出法をベクトル化によって高速化するのは難しい。具体的にはマクロブロックの1辺の長さ16しかベクトル長がとれない。そこでもとのエンコーダでの動きベクトルの探索の処理順序を変更し、更に探索範囲内の画素値の格納方式を変える。これらの変更によって、距離計算のときのベクトル長を大幅に長くすることにより、動きベクトル探索の高速化を実現する。

動きベクトルの探索の処理順序の変更について述べる。もとのエンコーダの処理では探索順序に関するループが外側にあり、探索範囲内の$16 \times 16$の画素領域に関するループが内側にある。このループを入れ換えることによって、ベクトル長を16から$2L+1$(探索範囲$L$が40のとき、81)にして動きベクトルの探索の高速化をはかる。しかしこれだけでは探索範囲が40のときでも、距離計算のベクトル長が81しかとれず、ベクトル計算機上で十分な性能は得られない。更に処理順序を変更しても距離計算のときの画素データのアクセスの種類が3種類あり、これもベクトル計算機上での高速化することにとって大きな支障となる。最初に図4の(A)のように行方向に長さ$2L+1$だけ連続に画素データがアクセスされる。その次に行が変わると図4の(B)のように画素データのアクセス位置が飛ぶ。最後に第1画素に関する距離計算から第2画素に関する距離計算に移るとき、図4の(C)のように画素データのアクセス位置が飛ぶ。

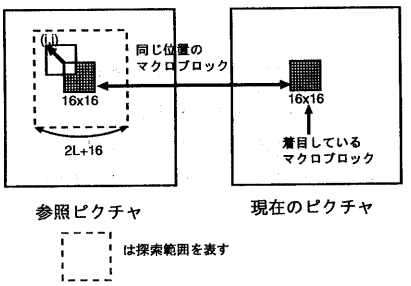



図3: 動きベクトルの検出法

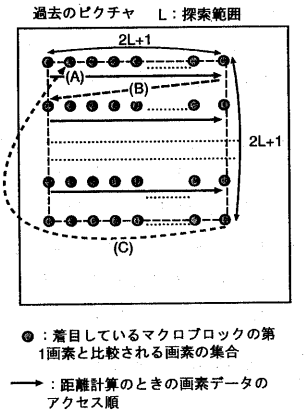



図4: 着目しているマクロブロックの第1画素と比較される画素の距離計算のときのアクセス順

このように不連続なメモリアクセスが生じるため、性能向上を考える上で大きな問題となる。これを解消するため、探索範囲内の画素値の格納方式を変更した。まず今回開発した距離計算手法において使う配列org_tmp[256][$(2L+1)^2$]を用意する。そして距離計算を行うとき、画素データが連続アクセスされるように、着目しているマクロブロックの各画素と比較される画素集合を配列org_tmpに格納していく(図5)。

ベクトル計算機上での距離計算は、この配列 org_tmp[256][$(2L+1)^2$] を使う。この配列を使うことによって、距離計算のとき画素データのアクセスが連続アクセスになり、距離計算の際のベクトル長は $(2L+1)^2$ となる。例えば L が 40 のときはベクトル長は 6561 になり、ベクトル長が短いという問題点と画素データのアクセスに不連続が生じるという問題点は解消される。これにより画素値のコピーというオーバーヘッドがあるにも関わらず、ベクトル計算機上で、距離計算が高速に実行されることが期待できる。

動きベクトル探索に関し、今回開発したベクトル計算機向きの手法と通常の手法について NEC 製 SX-4 で性能評価した。図 6 に探索範囲が 10、20、30、40 と変化した場合のもとの動きベクトル探索の CPU time(Original) と今回開発したベクトル計算機向きの動きベクトル探索の CPU time(Vectorized) を比較した結果を示す。但し、いずれの手法も SX-4 上でベクトル実行させて、性能評価を行った。

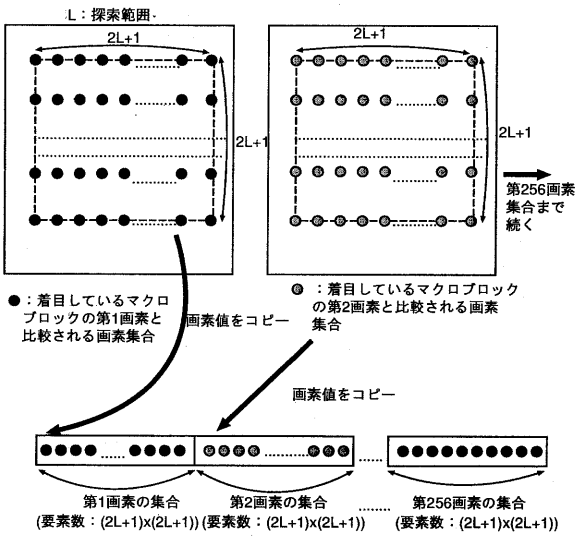



図 5: 画素データの格納方式の変更

ベクトル化による動きベクトル探索の速度向上の解析 (図 6)

- 画素値の格納方式を変えるというオーバヘッドがあるにも関わらず、動きベクトルの探索範囲が 10 と小さくても、開発したベクトル計算機向きの動きベクトルの探索法は、もとの探索法と比較し、2.82 倍の高速化を実現した。
- 動きベクトルの探索範囲が大きくなるほど、開発したベクトル計算機向きの動きベクトルの探索法は、距離計算のベクトル長が長くとれるため、もとの探索法に比べ高速に実行される。実際探索範囲が 10 のときは、2.82 倍の高速化であったのに対し、探索範囲が 40 のときは、6.12 倍の高速化を実現した。

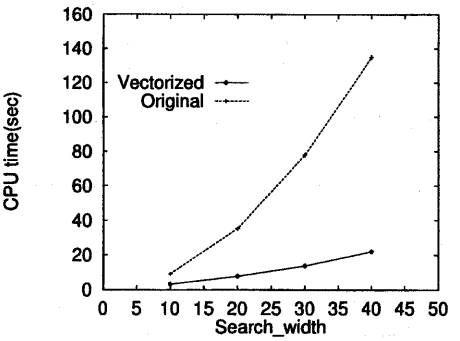
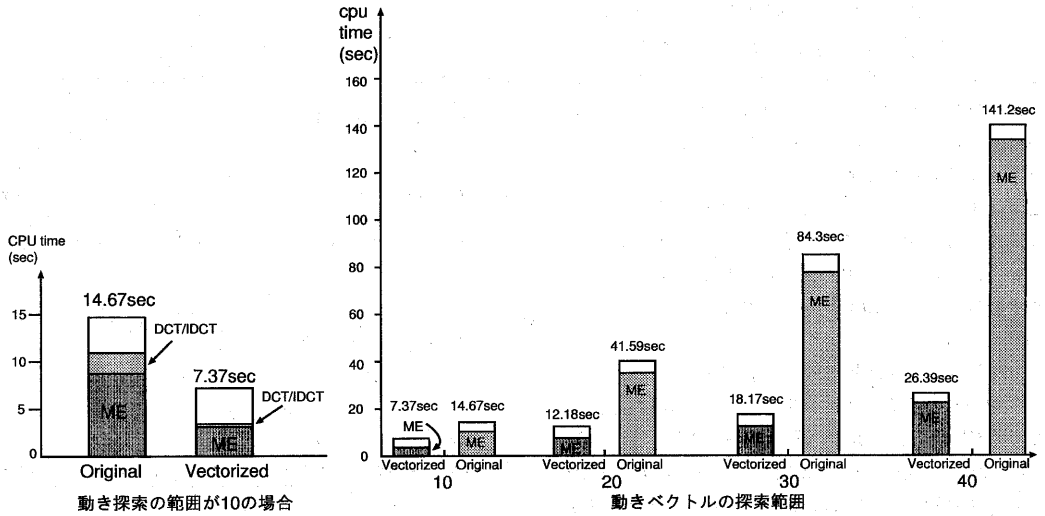



図 6: 探索範囲と動きベクトルの探索の CPU time の関係

## 5 MPEG-2 エンコーダ全体の高速化

本章では MPEG-2 エンコーダ全体に関し、今回開発したベクトル計算機向きの手法 (Vectorized) ともとの手法 (Original) を SX-4 上でベクトル実行させて、性能評価を行った。図 7 の (a) に探索範囲が 10 の場合、開発したベクトル計算機向きのエンコーダともとのエンコーダそれぞれについて ME(動きベクトルの探索)、DCT/IDCT の CPU time を比較した結果を示す。図 7 の (b) に探索範囲が 10、20、30、40 と変化したとき、もとの MPEG-2 エンコーダの CPU time と今回開発したベクトル計算機向きの MPEG-2 エンコーダの CPU time を比較した結果を示す。




(a) 探索範囲が 10 の場合の MPEG-2 エンコーダ内の ME,DCT/IDCT の CPU time

(b) MPEG-2 エンコーダの CPU time と動きベクトルの探索範囲の関係

図 7 : MPEG-2 エンコーダの SX-4 上での実行状況

SX-4 上での MPEG-2 エンコーダの実行状況の解析 (図 7)

- 動き探索の範囲が大きくなるにつれ、ベクトル計算機向きの手法は有効になる 。実際探索範囲が 10 の場合、もとのエンコーダの SX-4 上での性能に比べ、ベクトル計算機向きの手法の SX-4 上での性能は 2.0 倍であったのに対し、探索範囲が 40 の場合は 5.35 倍となる。

## 6 まとめ

本研究では MPEG-2 ソフトウェアエンコーダのうち、処理の核となる DCT/IDCT と動きベクトルの探索法のベクトル計算機向きアルゴリズムを検討し、その性能評価を述べた。 DCT/IDCT と動きベクトル探索それぞれについて、開発したベクトル計算機向きの手法をオリジナルのプログラムと比較し、 SX-4 上でそれぞれ約 18 倍と約 6 倍の高速化を実現した。更に SX-4 上で、開発したベクトル計算機向きの MPEG-2 エンコーダは、オリジナルのエンコーダと比較し、 5.35 倍の高速化を実現した。

今回開発した MPEG-2 ソフトウェアエンコーダのベクトル計算機アルゴリズムは、当グループが開発しているリアルタイム可視化システム RVSLIB の動画生成ツールに組み込む予定である。これにより、動画化された大規模数値計算結果を MPEG-2 方式で圧縮し、手軽に通信や蓄積することを目指している。

今後は、並列ベクトル計算機上での MPEG-2 ソフトウェアエンコーダの並列化・高速化を検討していく。更に数値シミュレーション画像の特質を生かし、画質、圧縮率、速度的により高性能な画像圧縮のアルゴリズムも検討していきたい。

## 参考文献

[1] S. Doi, et al.: *NEC Research & Development*, Vol.37, No.1, (1996) 114-123.

[2] B.G. Haskell, et al.: *Digital Video : An Introduction to MPEG-2*, Chapman & Hall, (1997).

[3] W. Chen, et al.: *A Fast Computational Algorithm for the Discrete Cosine Transform*, IEEE Trans. COM-25, (1977). 1004-1011.