I-014

マルチ GPU 上での画像圧縮における離散ウェーブレット変換の階層的並列処理

Hierarchical Parallelization of Discrete Wavelet Transform for Image Compression on Multi-GPU

吉川 一輝 †　高平 真由 †　吉田 明正 †
Kazuki Yoshikawa　Mayu Takahira　Akimasa Yoshida

1　はじめに

画像圧縮において高画質かつ高圧縮を実現するために，離散ウェーブレット変換が有効と考えられている．ウェーブレット変換に関する研究としては，JPEG2000 のウェーブレット変換 [1]，離散ウェーブレット変換の GPU 実装におけるメモリ使用量削減 [2]，ウェーブレット変換に基づいたチェックポイントデータの非可逆圧縮 [3]，1GPU を対象として画像圧縮におけるウェーブレット変換の並列処理 [4] 等がある．

本稿では，Daubechies 基底を用いた多重解像度解析 [5] に着目し，複数 GPU 上で並列処理を行うことにより高速化を実現する．本手法では，離散ウェーブレット変換を並列化する際に，対象画像の行レベルおよび列レベルの並列性を階層的に利用し，加えて，GPU 上のシェアードメモリを効果的に利用する画像分割法を提案する．性能評価においては，4K，8K，16K の画像データを使用し，NVIDIA Tesla K80 を 2 台（GK210 を計 4 台）用いて並列処理を行ったところ，高い実効性能が達成され，提案手法の有効性が確認された．

2　離散ウェーブレット変換による画像圧縮

本章では，離散ウェーブレット変換を用いたビットマップ画像の圧縮方法について述べる．画像圧縮では最初にビットマップ画像を輝度 Y，色差 Cb，Cr に変換する．次に画像に対して二次元のウェーブレット変換を行う．本手法ではこのウェーブレット変換においてマルチ GPU を使用した並列化を実現する．その後，求められた値から閾値以下の展開係数を 0 にしてから符号化を行い，画像圧縮の一連の処理が完了する．

2.1　離散ウェーブレット変換

離散ウェーブレット変換には，Haar のウェーブレット，Ingrid-Daubechies のウェーブレット，Cohen-Daubechies-Feauveau のウェーブレット等が存在するが，本稿では Ingrid-Daubechies のウェーブレット [5] を用いた．Ingrid-Daubechies のウェーブレットでは，予め用意された数列 p_k と q_k（N=10）を用いた．レベル $j(j >= 1)$ のスケーリング係数 $s_k^{(j)}$ と，ウェーブレット展開係数 $w_k^{(j)}$ は，式 (1) と式 (2) により求める．

$$s_k^{(j)} = \sum_n \overline{p_{n-2k}} s_n^{(j-1)} \tag{1}$$

$$w_k^{(j)} = \sum_n \overline{q_{n-2k}} s_n^{(j-1)} \tag{2}$$

†明治大学総合数理学部ネットワークデザイン学科
School of Interdisciplinary Mathematical Sciences, Meiji University

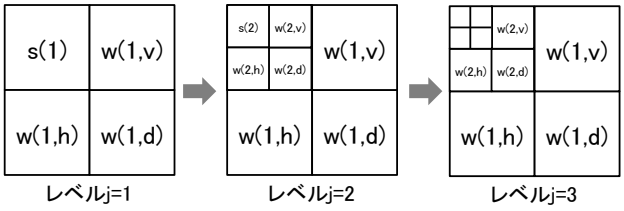


図 1　多重解像度解析．

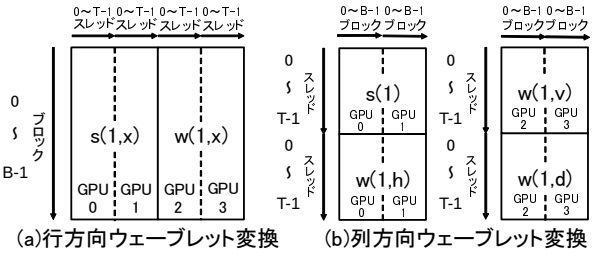


図 2　レベル j=1 のウェーブレット変換における GPU ブロック・スレッド割り当て．

2.2　多重解像度解析

本節では，本稿で用いた多重解像度解析の様子（レベル j=1〜3）を図 1 に示す．入力画像はレベル j=0 であり，s(0) とする．レベル j=1 において，w(1,v) には垂直方向の高周波成分，w(1,h) には水平方向の高周波数成分，w(1,d) には対角方向の高周波成分がそれぞれ現れるのが分かる．また，s(1) には s(0) を平均化した低周波成分が現れる．レベル j=2 はそれをさらに 4 分割，そしてレベル j=3 はレベル 2 のデータをさらに 4 分割した形となる．

3　マルチ GPU 上での離散ウェーブレット変換の階層的並列処理

本章では，離散ウェーブレット変換を GPU 上で階層的に並列処理する方法を述べる．

3.1　マルチ GPU 上での CUDA による並列化

提案するウェーブレット変換の階層的並列処理を GPU 上で実現するため，CUDA による並列プログラムを作成した．CUDA プログラムの各カーネル関数は，複数ブロック（MP で実行）とブロック内複数スレッド（MP 内 CUDA コアにより実行）を用いて階層的に並列処理される．本手法では，1 次元ウェーブレット変換，本手法では転置転送のオーバーヘッドを軽減するため，計算結果の保存先を転置後の場所 (メモリ) として，転置転送を除去している．1 次元ウェーブレット変換のカーネルにおけるブロック・スレッド割当てを 3.2 節で述べる．

3.2 ウェーブレット変換のブロック・スレッド割当て

基本的な多重解像度解析の各レベル j における 2 次元ウェーブレットの過程を，図 2 の (a)，(b) に示す．各 GPU でのカーネル実行の際にブロック数を B，スレッド数を T と設定する．レベル j=0 の s(0) からレベル j=1 の (a) を求める計算では，(a) の各行を GPU のブロック (0〜B-1) に割り当て，さらに (a) の各行の要素を GPU のブロック内スレッド (0〜T-1) に割り当てる．本手法では 4 つの GPU が使用可能であるため，画像を縦向きに二分割し，分割した画像の各行の要素をブロック内スレッド (0〜T-1) に割り当てて計算する．

次に (b) の各列を GPU のブロック (0〜B-1) に割り当て，さらに (b) それぞれの各列の要素を GPU のブロック内スレッド (0〜T-1) に割り当てる．

3.3 シェアードメモリサイズを考慮したウェーブレットデータの分割

本手法ではさらなる高速化を実現するために，ウェーブレット変換の対象となるデータを一般的に高速なシェアードメモリに転送してデータアクセス時間を短縮する．それぞれのウェーブレット対象データがシェアードメモリサイズ (48KB) に収まるように適切に分割を行っている．例えば，16K 画像の行方向ウェーブレット変換では 2 分割，列方向ウェーブレット変換では 4 分割している．

4 マルチ GPU 上での階層的並列処理の性能評価

本章では，Tesla K80 上で行ったウェーブレット変換の性能評価について述べる．

4.1 NVIDIA Tesla K80 搭載サーバ

性能評価に用いる DELL PowerEdge R730 は，GPU：NVIDIA Tesla K80，CPU：Intel Xeon E5-2680 v3 2.5GHz 12core*2，メモリ：64GB，OS：CentOS6.9，CUDA の処理系：CUDA Toolkit 8.0 となっている．

NVIDIA Tesla K80[6] は，Kepler アーキテクチャの GPU（GK210）を 2 基搭載した構成となっており，各 GPU は 2496CUDA コア（最大周波数 824MHz），12GB のデバイスメモリから構成されている．本性能評価では，K80 を 2 台 (GK210 を 4 基) 使用しており，各 GPU では 13MP(Multiprocessor)*192CUDA コア=2496CUDA コアを用いて並列実行を行った．

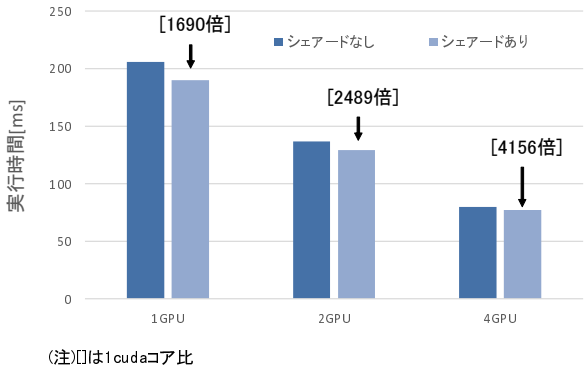
4.2 マルチ GPU 上でのウェーブレット変換の並列実行

本性能評価では，画像圧縮の対象となる 24 ビットカラーのビットマップ画像として，3 種類の画像データ（4K: 4096*4096，8K：8192*8192，16K：16384*16384）を用意した．性能評価においては，画像圧縮のうち GPU 上で行った Y（輝度）成分におけるウェーブレット変換の時間のみを測定している．GPU 実行におけるブロック数は 32，ブロック内のスレッド数は 1024 として実行時間を測定した．

実行結果を表 1 に示す．4K 画像の 1CUDA コアによる実行時間は 321,027[ms] であるのに対し，シェアードメモリを用いた並列実行時間は 4GPU で 77[ms] であり，図 3 のように 4156 倍の速度向上が得られている．同様に，8K 画像の場合は 1CUDA コアによる実行時間は 1,284,051[ms] であるのに対して，4GPU の並列実行時間は 335[ms] であり，3834 倍の速度向上が得られている．最後に，16K 画像の 4GPU による並列実行時間は

表 1　マルチ GPU 上でのウェーブレット変換の並列処理.

画像サイズ	シェアード	実行時間 [ms]			
		1CUDA コア	1GPU	2GPU	4GPU
4K	なし	321,027	206	137	80
	あり	-	190	129	77
8K	なし	1,284,051	1,064	664	384
	あり	-	873	547	335
16K	なし	-	4,106	2,550	1430
	あり	-	3,426	2,151	1160



(注)[]は1cudaコア比

図 3　4K 画像における GPU 実行時間.

1,160[ms] であり，高い並列実行性能が確認できた．いずれの場合も，シェアードメモリの利用により，実行時間が大幅に短縮されており，16K 画像の 4GPU 実行では 18.8%の実行時間短縮となっている．以上の結果，提案手法はマルチ GPU 上で高い実行性能を達成できることが確認された．

5 おわりに

本稿では，マルチ GPU における画像圧縮に用いられる離散ウェーブレット変換の階層的並列処理手法を提案した．本手法はマルチ GPU に対して適切な計算分散を行い，GPU のシェアードメモリを利用するためのデータ分割を行っている．NVIDIA Tesla K80 上で行った性能評価の結果から，4K サイズ〜16K サイズの画像に対して，十分な速度向上が得られており，提案手法の有効性が確認された．

参考文献

[1] 井上昂治, 江口翔馬, 黒木祥光, 黒崎正行, 尾知博．複数の GPU を用いたデジタルシネマ画像の実時間ウェーブレット変換．社団法人電子情報通信学会,2011.

[2] 生澤拓也, 伊野文彦, 荻原兼一．離散ウェーブレット変換の GPU 実装における入力の上書きによるメモリ使用量削減．情報処理学会研究報告,2015-HPC-148-10,2015.

[3] 佐々木尚人, 佐藤健斗, 遠藤敏夫, 松岡聡．実アプリケーションにおけるウェーブレット変換に基づくチェックポイントデータの非可逆圧縮手法．情報処理学会研究報告,2014-HPC-145-7,2014.

[4] 高平真由, 吉田明正．GPU 上でのウェーブレット変換を用いた画像圧縮の階層的並列処理．情報処理学会第 15 回情報科学技術フォーラム,I-032,2016/9.

[5] 中野宏毅, 山本鎮男, 吉田靖夫．ウェーブレットによる信号処理と画像処理．共立出版株式会社,1999.

[6] NVIDIA. TESLA K80 GPU ACCELERATOR, https://images.nvidia.com/content/pdf/kepler/Tesla-K80-BoardSpec-07317-001-v05.pdf,2015.