ノート

GPU による倍精度浮動小数点データの高速圧縮

山口　隆志*1)　　大原　衛*2)

Fast data compression of double-precision floating point numbers on GPU

Takashi Yamaguchi*1)，　Mamoru Ohara*2)

キーワード：GPGPU，倍精度浮動小数点数，可逆圧縮，数値シミュレーション
Keywords：GPGPU, double-precision floating point number, lossless compression, numerical simulation

1. まえがき

描画処理に特化されたハードウェアである GPU（Graphics Processing Unit）は，多数のプロセッサコアを内蔵しているため複数の処理を並列に実行することができる。そのため，高い演算能力を必要とする電磁界や流体，音響の数値シミュレーションや，リアルタイム性を要求される動画や画像の処理において GPU が用いられている。

GPU を搭載するビデオカード（以下デバイスと呼ぶ）は PCI Express バスを介してマザーボードと接続される。PCI Express の通信速度はデバイス内部の通信速度と比較して 1/10 程度と非常に遅い。したがって，デバイス上のメモリとマザーボード上のメモリ間や複数のデバイス間におけるデータ転送が GPU を用いた高速化における障害となる。特に，倍精度浮動小数点数を扱うことが多い数値シミュレーションでは転送するデータ量が非常に多くなるためその影響を受け易い。そこで本研究では，デバイス外部のバスを介した通信負荷を下げることを目的として倍精度浮動小数点数の圧縮手法について検討した。

2. 圧縮手法

2. 1　倍精度浮動小数点数の形式　　IEEE 754 規格に従った倍精度浮動小数点数は図 1 のような 64 ビットのデータで表される。図 1 において，s は符号ビットであり exp は指数部，$fraction$ は仮数部である。本研究では，符号ビット s を 1 ビット目として図 1 の右に向かって 2 ビット目，3 ビット目と数える。これらの表記を用いると任意の実数 f は

$$f = (-1)^s \times 2^{(exp-1023)} \times (1.0 + fraction) \quad \text{.....(1)}$$

で与えられる。指数の符号ビットを省略するため，実際の値に 1023 を足したものが exp に保存される。また，仮数の整数部は常に 1 であるとして正規化されており fraction は小数部のみを表している。

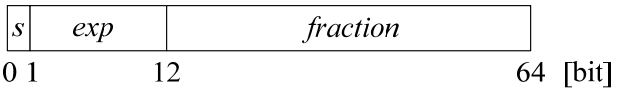

図 1.　IEEE 754 倍精度浮動小数点形式

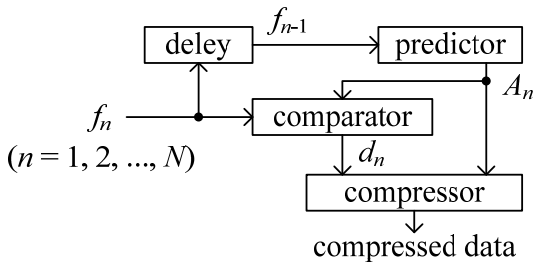

図 2.　FPC のブロック図

2. 2　従来手法　　倍精度浮動小数点数の圧縮手法として FPC が報告されている[1]。FPC のブロック図を図 2 に示す。FPC による圧縮では，以前に入力された値から f_n に対する予測値 A_n を算出し，A_n と f_n の排他的論理和 d_n を導出する。予測の精度が高いほど d_n のビットパターンには多くの 0 が含まれる。f_n より少ない情報量で d_n を表現することができれば圧縮が可能となる。圧縮データを伸長する際には，既に伸長されたデータから A_n を算出できるため，d_n の情報だけが得られればよい。

FPC の予測器は，異なる 2 種類の予測アルゴリズム[2][3]によって $A_{n,1}$ と $A_{n,2}$ を算出し，精度が高い方を採用する方法を用いている。しかし，いずれの予測アルゴリズムも逐次型の計算を行うため GPU による高速化が不可能である。また，プロセッサ内部で利用される予測のために設計されたものであるため，物理量を表す数値シミュレーションのデータに必ずしも適していないという問題がある。

d_n は，先頭ビットから 0 が連続するバイト数 L を表 1 の 3 ビット符号に置き換えることにより圧縮される。ただし，経験的に生起確率の低い $L = 4$ には符号を与えず $L = 0$ として扱う。$L + 1$ バイト目から 8 バイト目までは何も操作せず

*1) 情報技術グループ
*2) 経営企画室

表 1. 連続する 0 の数を表す符号表

L	0	1	2	3	4	5	6	7	8
code	000	001	010	011	-	100	101	110	111

そのまま出力する。また，複数の予測器を用いる場合には，表 1 の符号に 1 ビットの選択ビットを付加したものを出力する。したがって，8 バイトの入力 f_n が 1/2 + (8 – L)バイトに圧縮されることになる。

2. 3 提案手法 GPU による数値シミュレーションにおいて有効な倍精度浮動小数点数の圧縮を行うために多項式補外を用いた予測器（PEP）について検討する。本研究では，$m + 1$ 個の点$(x_i, f(x_i))$, $(x_{i+1}, f(x_{i+1}))$, …, $(x_{i+m}, f(x_{i+m}))$を通る次式の m 次 Lagrange 補間多項式[4]を用いる。

$$P(x) = \sum_{j=i}^{i+m} \frac{\prod_{k=i}^{i+m}{}'(x - x_k)}{\prod_{k=i}^{i+m}{}'(x_j - x_k)} \cdot f(x_j) \quad \cdots\cdots (2)$$

ただし，Π' は $j = k$ の項を除いた積を表す。数値シミュレーションで扱う物理量は波のように変化する連続量である場合が多い。したがって，式（2）による近似で精度よく予測できると考えられる。また，予測するために必要なデータ数が $m + 1$ 個に限られるため，複数の予測値を並列して計算することができる。

表 2 は，5 つのデータセットについて PEP を用いた場合における L を示したものである。表中の P_0 & $P_i (i = 1, 2, 3, 4)$ は，0 次（直前値）と i 次の PEP 出力のうち良好な方を採用するハイブリット方式の結果を表す。データセット brain と comet，control，plasma は文献（1）で扱われている数値シミュレーションのデータであり，elemag は著者らが作成した finite-difference time-domain 法による電磁界シミュレータの出力データである。最長の L を得る予測器の組み合わせはデータセットによって異なることが表より分かる。

3. 実装結果

表 2 に示した予測器の組み合わせで圧縮処理を行った結果を表 3 に示す。各行の上段は圧縮後のデータサイズを表しており，下段は元データのサイズに対する比（圧縮率）を表している。表 3 より，plasma を除くすべてのデータセットにおいて圧縮率は 1 以下であることが分かる。特に，電磁波の空間分布データである elemag では，予測器を P_0 & P_3 とすることで元のデータサイズの 3/4 に圧縮することが可能である。細かく振動している数値データである plasma の場合，式（2）による予測精度が低いため圧縮率が 1 を超える結果になったと考えられる。

各データセットにおいて比較的良好な圧縮率となった P_0 & P_2 を使用した場合の圧縮に要する時間を表 4 に示す。GPU は，240 個のプロセッサコアを持つ Tesla C1060 を用いた。表 4 の 1 列目は，同時に実行したスレッド数（並列数）を表している。並列数を上げることによって処理時間が短くなっていくことが表 4 より分かる。式（2）による予測値は

表 2. 連続する 0 の数を表す符号表

Predictor	brain	comet	control	plasma	elemag
P_0 & P_1	1.51	1.60	0.96	0.49	2.26
P_0 & P_2	1.46	1.63	1.04	0.49	2.43
P_0 & P_3	1.43	1.64	1.07	0.49	2.51
P_0 & P_4	1.41	1.64	1.09	0.49	2.47

表 3. 圧縮後データサイズ [Mbyte]

Predictor	brain	comet	control	plasma	elemag
P_0 & P_1	123.92 (0.874)	92.59 (0.862)	150.36 (0.943)	35.14 (1.001)	852.35 (0.781)
P_0 & P_2	124.78 (0.880)	92.10 (0.858)	148.86 (0.933)	35.14 (1.001)	829.12 (0.760)
P_0 & P_3	125.39 (0.884)	92.08 (0.858)	148.10 (0.928)	35.14 (1.001)	818.66 (0.750)
P_0 & P_4	125.78 (0.887)	92.12 (0.858)	147.89 (0.927)	35.14 (1.001)	823.27 (0.755)

（）内の数値は圧縮率を表す。

表 4. 処理時間（P_0 & P_2）

Threads	Time [ms]				
	brain	comet	control	plasma	elemag
16	2207	1671	2481	547	16994
32	1202	911	1349	298	9287
64	731	552	819	181	5616
128	604	457	680	150	4645
256	314	239	354	79	2413
512	234	180	270	54	1788

並列数を上げても変わらないため，スレッドの数を増やしても圧縮率に影響はなく表 3 と同じ結果になる。

4. まとめ

GPU を用いた数値シミュレーションにおいて低速バスを介した通信負荷を下げるために倍精度浮動小数点データの圧縮について検討した。予測器に多項式補外を用いることで高速に圧縮処理できることが分かった。今後の課題として，予測器出力のデータフォーマットを改良することによる圧縮率の向上や，多項式補間以外の予測器とハイブリット化することによる振動データへの対応などがある。

（平成 23 年 5 月 17 日受付，平成 23 年 7 月 1 日再受付）

文 献

(1) M. Burtscher and P. Ratanaworabhan: “FPC: A High-Speed Compressor for Double-Presicion Floating-Point Data”, IEEE Trans. Comput., Vol. 58, No. 1, pp. 18–31 (2009)
(2) Y. Sazeides and J. E. Smith: “The Predictability of Data Values”, Proc. IEEE/ACM 30th Int’l Symp. Microarchitecture (MICRO’97), pp. 248–258 (1997)
(3) B. Goeman, H. Vandierendonck, and K. Bosschere: “Differential FCM: Increasing Value Prediction Accuracy by Improving Table Usage Efficiency”, Proc. 7th Int’l Symp. High Performance Computer Architecture (HPCA’01), pp. 207–216 (2001)
(4) 戸川隼人：「数値計算技法」，オーム社 (1972)