Vol.2014-HPC-143 No.4
2014/3/3

時間発展シミュレーション向けの時系列データ圧縮手法

松尾 勇気[1]　石川 裕[1]

概要：時間発展シミュレーションは定期的に計算結果を出力するため、巨大なデータが生成される。そのデータサイズを縮小するため、データ圧縮技術が使われているが、ある時間ステップにおいて書き出すデータの隣接する値の類似性を活用している。本研究では、予測器に基づいた既存の高速な浮動小数点圧縮アルゴリズムを基にした時系列データ圧縮器 t-FPC を提案する。圧縮率を向上させるため、1 つの時間ステップに出力される中間データを、同一の時間ステップのデータ内部のものではなく、最近の複数の時間ステップで出力されたデータ間の差分を用いて圧縮を行う点でそれとは異なる。さらに、書き出す差分データ全体の bit 長の分布から可変サイズでデータを書き出すことにより圧縮率を向上する。また、連続して同じ値の場合、時間ステップの出力が全て同じ値の場合には、特別なエンコーディングをすることにより、圧縮率をさらに向上させる。時間発展シミュレーションである SCALE 気象・気候モデルの実際のファイル出力データを用いた評価により、t-FPC は、既存の浮動小数点データ専用の圧縮器や一般のデータ圧縮器に比べて、圧縮率や処理スループットの観点において、良い性能を達成していることを示す。

TIME-SERIES DATA COMPRESSION METHOD FOR TIME EVOLUTION SIMULATIONS

Yuki Matsuo[1]　Yutaka Ishikawa[1]

Abstract: Time evolution simulations generate a large amount of data periodically to ouput results. Existing compression techniques reduce data size by utilizing the similarity among the neighboring values inside the data that is written at one time step. In this paper, we propose t-FPC, a time-series data compression method that adapts the existing predictor based fast floating point compression algorithm. Compared to previous compression methods, it differs in the aspect that it compresses the intermediate result of one time step utilizing the finite differences among the data of recent multiple time steps, instead of those inside the data at the same time step in order to increase the compression ratio. In t-FPC, diffs are compressed with eight kinds of bit lengths. Those are determined based on bit length distribution of diffs. Special encoding is introduced for the two cases: one is that an element at one time step is the same as the previous time step, and another is that all elements at one time step is completely the same as the previous time step. Using the SCALE weather and climate model, it is shown that t-FPC achieves better processing speed and compression ratio than other data compressors including a famous floating point compressor.

1. はじめに

時間発展するシミュレーションプログラムなどの HPC アプリケーションでは定期的にシミュレーション結果をファイルに書き出し、そのデータを後の解析、あるいは視覚化に使用する。2018 年〜2020 年に登場するだろうと考えられているエクサスケールスーパーコンピュータ [1] では、より高精度のシミュレーションが行われるようになり生成されるファイルサイズも増大することが予想されている [2]。増大するデータに対してデータ圧縮技術はストレージを有効利用できるようになるだけでなく、要求する I/O バンド幅を減少させることも可能となる。

本論文では、浮動小数点データに対して予測器を用いて圧縮する FPC(Floating Point Compression) 手法 [3] に基づく t-FPC 手法を提案する。FPC 圧縮の特徴は、近傍 3 つのデータと直前のデータから圧縮データすべきデータの値を予測し、その予測との残余 (差分) を圧縮データとし

[1] 東京大学情報理工学系研究科

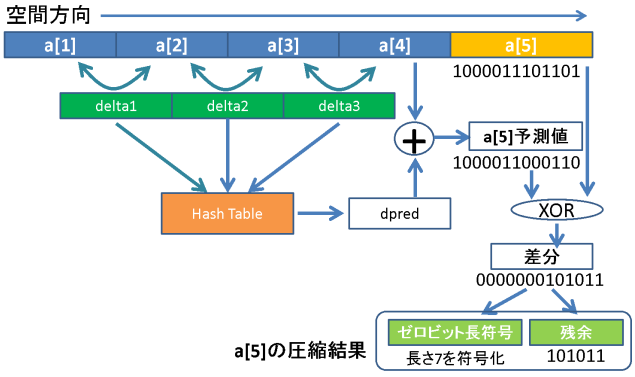

図 1 FPC の処理概要

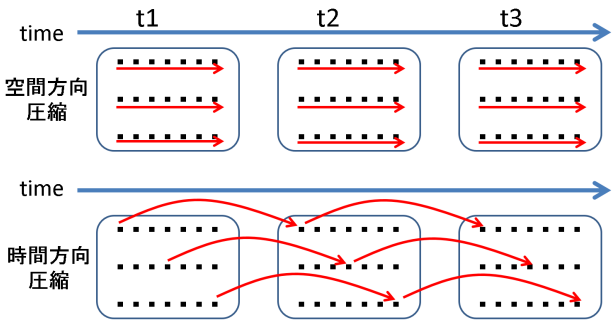

図 2 圧縮方向

て扱うところにある。t-FPC 手法は次の 4 つの特徴を持っている。1) FPC では空間方向のデータ圧縮を行なっているのに対し、t-FPC では時間方向のデータ圧縮を行う。2) FPC の残余データの長さは 1 バイトから 8 バイトのいずれかに固定されるのに対し、t-FPC では残余データのビット長の出現頻度から 8 種類のビット長を定義して使用する。これにより FPC よりも圧縮率が高くなる。FPC も t-FPC も 8 種類のデータ長を持つため圧縮データの先頭に 3 ビットのエンコーディング情報を付加している。3) あるデータが直前に書きだしたデータと同じ値の場合には、値が同じであるというフラグを設ける。このために通常圧縮データのエンコーディングフィールド 3 ビットにさらに 1 ビット追加している。4) 直前のタイムステップで書きだしたデータが全部同じ値であった場合の識別子をタイムステップ毎に記録している。これにより、時間発展するシミュレーションで、さほどパラメータの値が変わらないファイルに対しては高い圧縮が達成できる。

2. 設計

本章では、まず、提案する手法の元になった FPC(Floating Point Compression) 手法について紹介した後、時間発展型シミュレーションのデータ書き出し向け圧縮手法 t-FPC を設計する。

2.1 FPC

図 1 は、FPC 手法を用いて、配列 a に格納されている浮動小数点データを圧縮している様子を示している。ここでは浮動小数点データは 14bit で表現されているものとする。以下、圧縮手順を示す。

(1) 最初の 4 つのデータは圧縮することなくファイルにデータを書き出し、5 番目のデータから圧縮される。5 番目のデータを圧縮するまでに、a[1] と a[2]、a[2] と a[3]、a[3] と a[4] の残余（差分）が計算されている。

(2) これら 3 つの残余をハッシュキーとしてハッシュテーブルを引く。最初ハッシュテーブルは空なので、dpred の値は 0 となる。

(3) dpred と a[4] の値のビット和を a[5] の予測値とする。

(4) a[5] の実際の値と予測値のビット排他的和をとることにより残余が得られる。この残余は、先に使用したハッシュキーを使ってハッシュテーブルに格納される。

(5) IEEE754 形式の浮動小数点では、上位ビットから符号部、指数部、仮数部で構成される。データ値の差が小さい場合、符号部、指数部、仮数部の上位ビットの値は同じである。すなわち、残余が小さいほど上位ビットはゼロで埋められている。図の例では、14bit データ中 7bit がゼロとなっており、6bit 分を保存すれば良い。

(6) 一つの圧縮データはゼロビット長符号と残余から構成される。ゼロビット長符号ビットは言い換えると残余ビットの長さを表現している。FPC の場合、ゼロビット長符号は 3bit で表現されており、残余部分を 1 バイトから 8 バイトの 8 通り表現する。

時間発展するシミュレーションのタイムステップごとのデータ書き出し時に FPC 圧縮器を適用した場合、図 2 の空間方向圧縮をしていることになる。すなわち、各タイムステップ毎のデータ群（空間）を圧縮している。この空間が FPC がアクセスするメモリ領域に対して類似性があれば高い圧縮が望めるが、そうでない場合には低くなる。時間発展するシミュレーションの場合、空間内の類似性よりもタイムステップ間（時間方向）での類似性が高いことが予想される。t-FPC は、時間方向で類似性に着目するとともに FPC の欠点を解決した圧縮手法である。

2.2 t-FPC

FPC を時間方向圧縮に適用するためには、圧縮しているタイムステップの直前 3 つの残余を保持しておく必要がある。t-FPC ユーザに対しては、通常のデータ圧縮・解凍 API となるようにし、ライブラリの中で必要な履歴を保存するようにする。図 3 にデータ構造と API を示す。

図 4 に t-FPC の処理概要を示す。FPC 同様直近の 3 つのデータの残余を保持するが、FPC と違って残余データはタイムステップ毎に書きだされたデータ領域全体を保持することになる。それ以外の処理手順は FPC と同じである。

```
typedef struct {
    union double_uint* prev;
    uint64_t* delta[3];
    size_t nelm;
    int state;
    struct hash_entry* ht;
} struct_tfpc;

size_t compress(double* in, double* out, struct_tfpc* st);
void decompress(double* in, double* out, struct_tfpc* st,
                int flag);
```

図 3 t-FPC データ構造と API

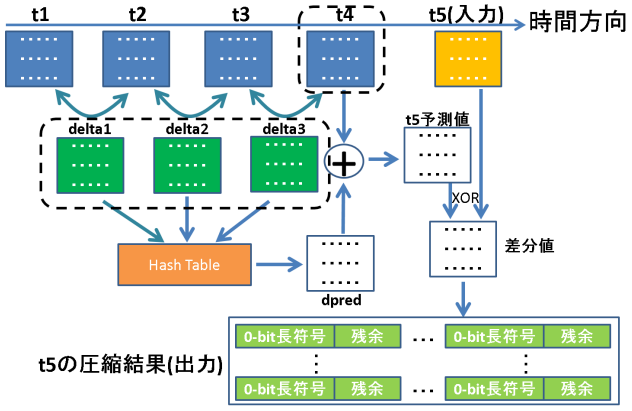

図 4 t-FPC の処理概要

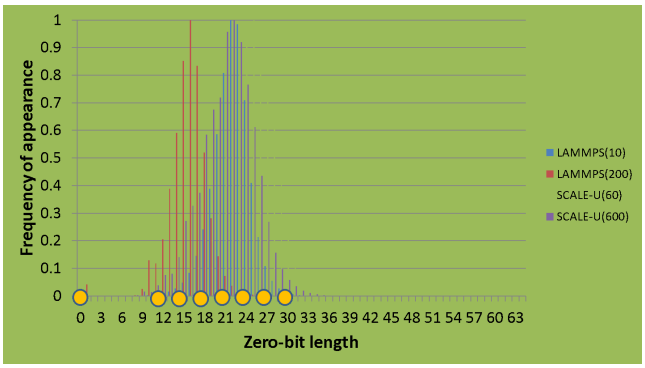

図 5 ゼロビット長出現頻度

2.2.1 ゼロビット長符号の最適化

FPC のゼロビット長符号では、残余データサイズがバイト単位となる。例えば、残余が 2bit しかなくても 8bit 分割り当てられるため効率が良くない。ゼロビット長符号は 3bit ある。それぞれの符号が何ビットに対応しているかは、残余の分散を見て決めることにより効率良い符号化が可能となる。

図 5 に Sandia National Laboratory で開発された分子動力学シミュレータ LAMMPS[4]、理研計算科学研究機構で開発された気象・気候モデル SCALE[5] におけるデータ圧縮のゼロビット長出現頻度を示す。出現頻度は同じ傾向を示しており、10bit から 30bit の間をきめ細かく分類すると効率が良いことが分かる。t-FPC では、このような符号化を行なっている。

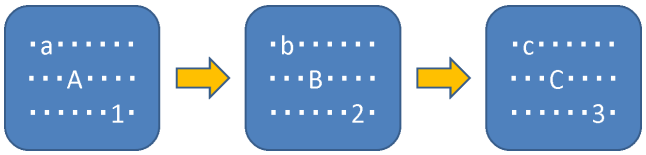

図 6 残余領域が少ない場合

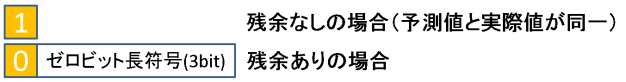

図 7 残余なしフラグ

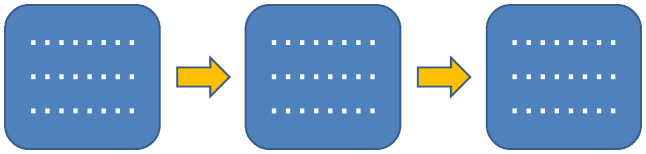

図 8 残余が同じ場合

2.2.2 残余領域が少ない場合の圧縮率向上

図 6 は、3 つのタイムステップでのデータ領域の内容を示している。点で示しているデータの値は変動せず、それ以外のデータの値が変動している。FPC の圧縮では、同じ値であっても圧縮データは最低限ゼロビット長符号の 3bit が必要となる。これを避けるために、図 7 に示す通り t-FPC ではゼロビット長符号の前に 1 ビットフラグを設ける。フラグが 1 の場合は同じ値、フラグが 0 の場合は残余があるために圧縮データが入る。

2.2.3 残余が同じ場合の圧縮率向上

図 8 は、3 つのタイムステップでのデータ領域全てが同じ値である場合を示している。この場合には、当該ステップは同じであるという識別子を設けることにより全体を圧縮する。

3. 評価

気象・気候モデル SCALE[5] を用いて、t-FPC を FPC、gzip[6], bzip2[6] と比較する。パラメータとして velocity(U, V, W)、potential temperature(PT)、relative humidity(RH)、total water(QTOT) を書き出している。また、書き出し間隔は 60 秒と 600 秒の 2 つのケースで評価している。書き出し間隔が長くなるとデータの類似度は下がり圧縮率が低下する。間隔によってどのくらい低下するかを調べた。

図 9 に圧縮率の結果を示す。gzip, bzip2 よりも t-FPC、FPC は高い圧縮率を達成している。60 秒間隔の書き出しでは、FPC に比べて 13.5 %の圧縮率向上、600 秒間隔の書き出しでは、FPC に比べて 2.3 %の圧縮率向上となった。

図 10 に圧縮および解凍時間の結果を示す。t-FPC の圧縮時間は FPC よりは時間がかかっているが gzip に比べ 66%時間が短縮している。

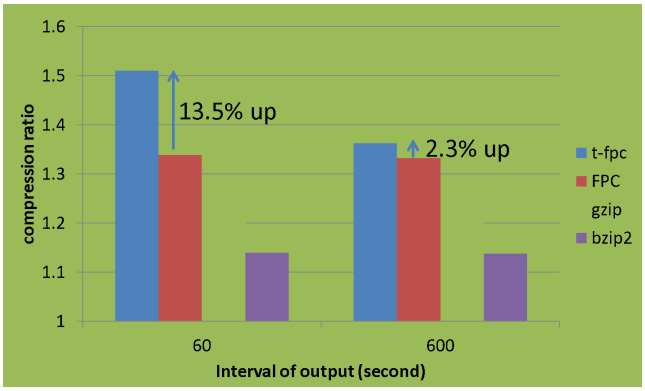

図 9 圧縮率

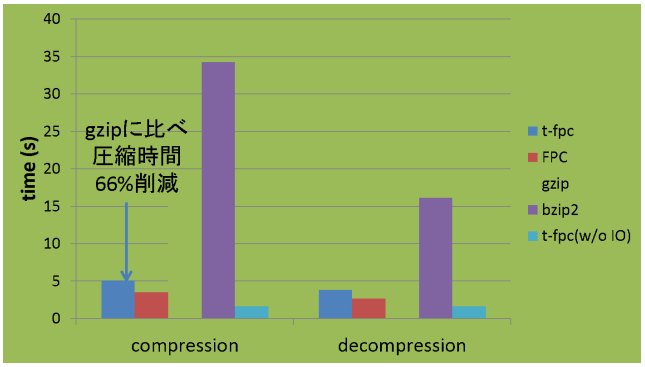

図 10 圧縮/解凍時間

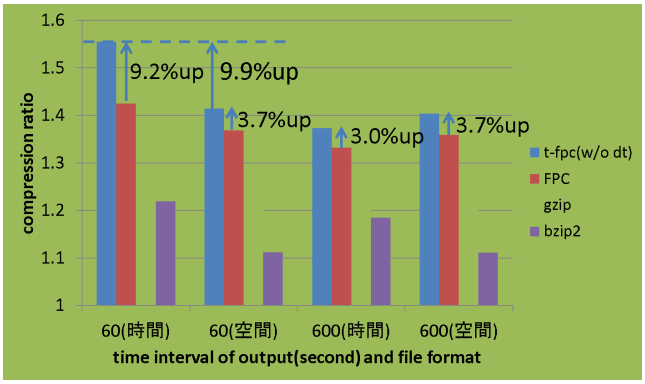

図 11 圧縮方向の影響

図 11 に空間方向、時間方向での圧縮率を比較した。いずれも t-FPC が高い圧縮率を達成していることが分かる。

4. 関連研究

Hogan ら [7] は、チェックポイント間のデータ差分を取り、それをさらに DEFLATE アルゴリズムで圧縮する手法を提案している。データ差分を取った時に差分値に閾値を設定し、閾値以下の場合にはゼロに丸めるという手法も提案している。論文 [8] では、データをマスクして情報エントロピーを下げた後に通常の圧縮器を使用することにより、より高い圧縮率が達成できることを報告している。論文 [9] では、 Lorenzo 予測器を用いた浮動小数点圧縮器を提案している。

5. おわりに

予測器に基づいた既存の高速な浮動小数点圧縮アルゴリズムを基にした時系列データ圧縮器 t-FPC を提案し評価した。t-FPC は時間発展シミュレーションの時間ステップ毎にデータを書きだすような時系列データに対して時間方向の圧縮を行う。

t-FPC は、既存研究 [3] を時間方向の圧縮に適用するとともに、データ圧縮に改良を加えた。これにより、t-FPC は FPC に比べて空間方向での圧縮においても高い圧縮率を達成している。今後の研究課題としては t-FPC の並列化がある。並列化により圧縮スピードを上げ、ファイル I/O 処理時間の短縮を図る。

謝辞

本研究の一部は、文部科学省「将来の HPCI システムのあり方の調査研究」課題名「レイテンシコアの高度化・高効率化による将来の HPCI システムに関する調査研究」、および、科学技術振興機構 CREST「科学的発見・社会的課題解決に向けた各分野のビッグデータ利活用推進のための次世代アプリケーション技術の創出・高度化」領域のなかの課題名「「ビッグデータ同化」の技術革新の創出によるゲリラ豪雨予測の実証」による。

参考文献

[1] Dongarra, J. and Beckman, P.: The International Exascale Software Roadmap, *International Journal of High Performance Computer Applications*, No. 1 (2011).
[2] : 計算科学ロードマップ 中間報告書.
[3] Goeman, B., Vandierendonck, H. and De Bosschere, K.: Differential FCM: increasing value prediction accuracy by improving table usage efficiency, *High-Performance Computer Architecture, 2001. HPCA. The Seventh International Symposium on*, pp. 207–216 (online), DOI: 10.1109/HPCA.2001.903264 (2001).
[4] : LAMMPS Molecular Dynamics Simulator, *http://lammps.sandia.gov/*, (online), available from ⟨http://lammps.sandia.gov/⟩.
[5] : SCALE Scalable Computing for Advanced Library and Environment, *http://www.gfd-dennou.org/arch/davis/workshop/2012-12-12/nishizawa_20121212.pdf*.
[6] : bzip2, *http://www.bzip.org*.
[7] Hogan, S., Hammond, J. and Chien, A.: An evaluation of difference and threshold techniques for efficient checkpoints, *DSN Workshops*, IEEE (2012).
[8] Gomez, L. A. and Cappello, F.: Improving foating point compression through binary masks, *International Conference on Big Data*, pp. 326–331 (2013).
[9] Lindstrom, P. and Isenburg, M.: Fast and efficient compression of oatingpoint data, Vol. 12, No. 5 (2006).

正誤表

図 5、9、10、11 は順に次が正

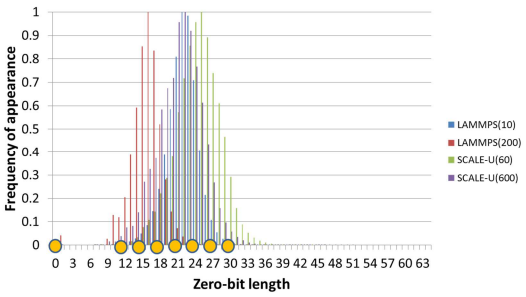

図 5 ゼロビット長出現頻度

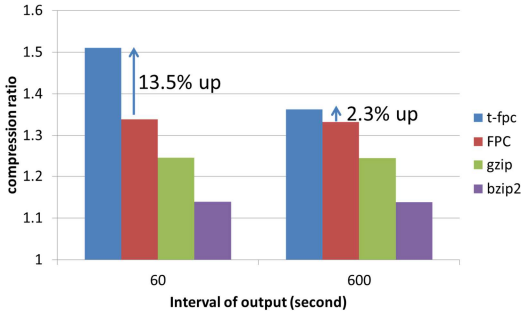

図 9 圧縮率

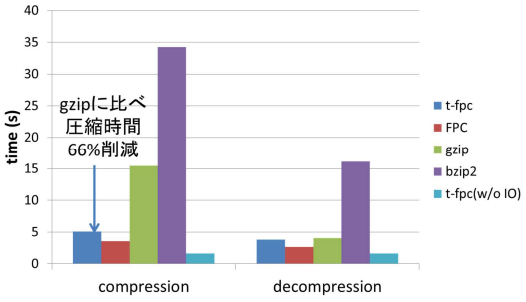

図 10 圧縮/解凍時間

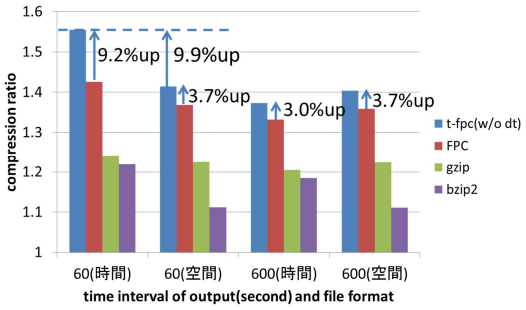

図 11 圧縮方向の影響

参考文献[3]は次の 2 文献の引用が正

[3-1] Burtscher, M. and Ratanaworabhan, P.: FPC: A High-Speed Compressor for Double-Precision Floating-Point Data, Computers, IEEE Transactions on, Vol. 58, No. 1, pp. 18–31 (online), DOI: 10.1109/ TC.2008.131(2009).

[3-2] Ratanaworabhan, P., Ke, J. and Burtscher, M.: Fast lossless compression of scientific floating-point data, Data Compression Conference, 2006. DCC 2006. Proceedings, pp. 133–142 (online), DOI: 10.1109/ DCC. 2006.35 (2006).

1 章

誤「あるデータが直前に書きだしたデータと同じ値の場合には」

正「同じ値が連続する場合には、正確な値予測が可能になる。あるデータが予測値と同じ場合には」

誤「直前のタイムステップで書きだしたデータが全部同じ値であった場合の識別子」

正「あるタイムステップで書き出したデータが全部予測値と同じ値であった場合の識別子」

2.1 節

誤「ここでは浮動小数点データは 14bit で表現されている」

正「ここでは浮動小数点データは 13bit で表現されている」

誤「図の例では、14bit 中」

正「図の例では、13bit 中」

図 6 の説明文

誤「残余領域が少ない場合」 正「データ変動が少ない場合」

図 8 の説明文

誤「残余が同じ場合」 正「データが変動しない場合」

2.2.2 節の表題

誤「残余領域が少ない場合の圧縮率向上」

正「データ変動が少ない場合の圧縮率向上」

2.2.3 節の表題

誤「残余が同じ場合の圧縮率向上」

正「データが変動しない場合の圧縮率向上」

2.2.3 節

誤「当該ステップは同じであるという識別子」

正「当該ステップの値は全て予測値と同一になり、そのことを表す識別子」