4K-7

AVX を用いた BCRS 形式疎行列ベクトル積の特性評価

佐藤真之介† 菱沼利彰† 藤井昭宏‡ 田中輝雄‡
工学院大学大学院情報学専攻† 工学院大学情報学部‡

1. はじめに

大規模シミュレーションでは，問題を離散化することにより得られる疎行列ベクトル積 $\boldsymbol{y} = A\boldsymbol{x}$ (SpMV)が用いられている(ここで $\boldsymbol{x}, \boldsymbol{y}$ はベクトル，A は行列を表す)．一般的に，SpMV ではメモリを節約し演算量を削減するため，疎行列を非零要素のみに圧縮して扱う．疎行列のデータ格納形式の一つとして圧縮行格納形式(CRS)[1]がある．また，疎行列の形状によっては，CRS のデータ格納形式をブロック化して保持するブロック圧縮行格納形式(BCRS)を用いることにより，SpMV を高速化することができる．しかし，BCRS 形式の効果は疎行列の形状に依存し，効果が限定的であった．

一方，近年，ストリーミング SIMD 拡張命令(SSE)と呼ぶ高速化機能が登場し，現在では同時に 4 つの倍精度浮動小数点演算が可能な Intel Advanced Vector Extensions(AVX)[2] が実用化されている．この AVX を利用した CRS 形式の SpMV についての研究が行われている[3].

本研究では，BCRS 形式によるブロック化と AVX の整合性に着目し，AVX を用いた BCRS 形式 SpMV について評価を行った．

2. 圧縮格納形式

CRS と BCRS のデータ格納形式を図 1 に示す．

まず，CRS 形式は疎行列の非零要素を行方向に格納する形式である．データを保持するために，非零要素の値(Val)，要素の列番号(Col_ind)，要素上での行番号の開始位置(Row_ptr)の3つの配列を用いる．次に，BCRS 形式は疎行列の非零要素をブロックに分割し格納する．ブロック単位で格納するため，ブロック内の零要素も非零要素と一緒に格納する．

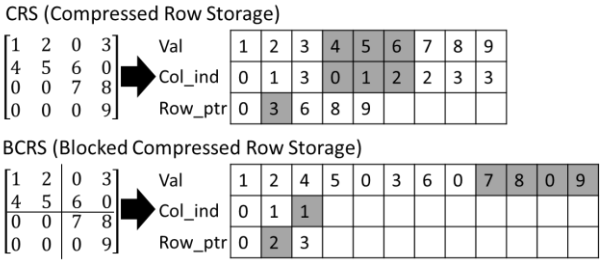

図 1 CRS と BCRS のデータ格納形式

BCRS 形式は，零を含むブロック内の要素(Val)，ブロック単位でみた列番号(Col_ind)，Col_ind 上でのブロック幅ごとの行番号の開始位置(Row_ptr)の 3 つの配列から構成されている．

3. AVX を用いた BCRS 形式 SpMV

AVX では 4 つの倍精度のデータに対して同時に浮動小数点演算ができる．

CRS 形式の SpMV の場合，ベクトル $\boldsymbol{x}$ は圧縮していない．そのため，非零要素が連続に確保されていない場合，AVX を利用するために，データを連続に再配置する必要がある．

一方，BCRS 形式の SpMV では，ブロック内は連続であるため(零要素が含む場合もある)，ブロックサイズを AVX の演算器構成と合わせれば，AVX を効率良く利用できる可能性がある．今回，BCRS 形式におけるブロックの行と列のサイズを，AVX の演算器構成を考慮して 1×4，4×1，4×4 の 3 パターンを用意した．なお，1×4 のパターンでは，ブロック内の計算結果はベクトル $\boldsymbol{y}$ の 1 つの要素になるため，リダクションを必要とする．4×1 と 4×4 のパターンでは，ベクトル $\boldsymbol{y}$ のそれぞれの要素となり，リダクションの必要がない．

4. 評価実験

実験には Intel Xeon corei7-3770K 3.5GHz(4core) を用いた．コンパイラは Intel C++ Compiler 12.0.3, オプションは”-O3，-openmp，-xAVX”を用いた．

Characterization of Sparse Matrix-Vector Multiplication in BCRS using AVX
Shinnosuke Sato†, Toshiaki Hishinuma† Akihiro Fujii‡ and Teruo Tanaka‡
† Graduate School of Infomatics, Kogakuin University
‡ Faculty of Infomatics, Kogakuin University

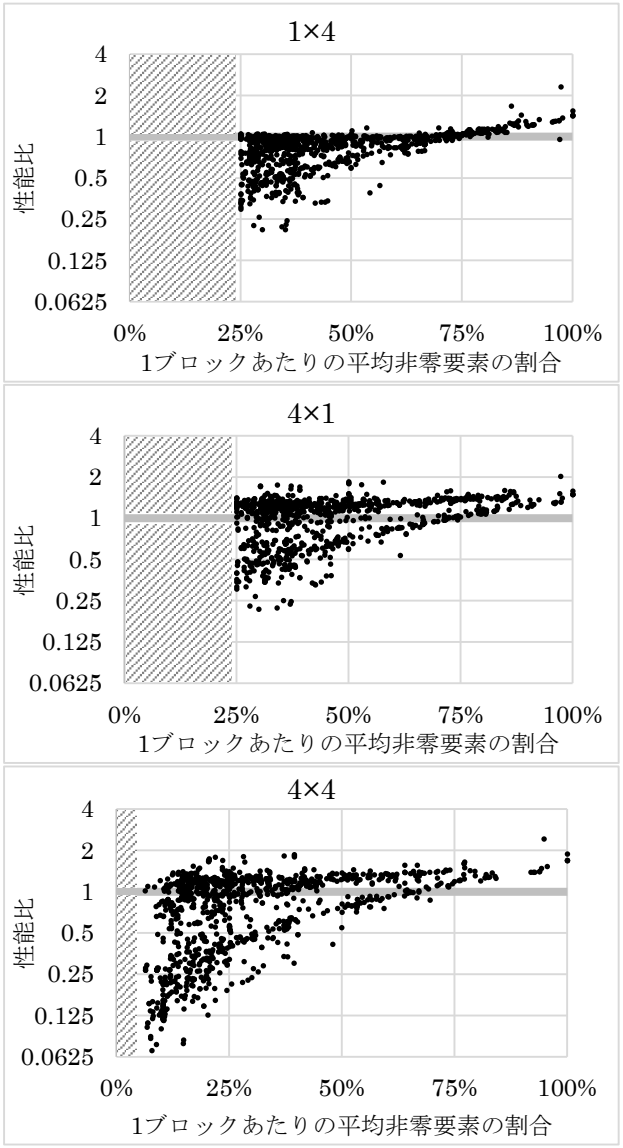

図 2　CRS を基準とした BCRS の性能比

実験には，The University of Florida Sparse Matrix Collection[4]に登録されている形状の異なる 1333 種類の実数の疎行列を用いた．これらの疎行列に対し，CRS 形式と 3 つのパターンの BCRS 形式について実測した．図 2 に CRS 形式を基準とした BCRS 形式の各パターンの性能比を示す. BCRS 形式の SpMV の性能は，全体の 73%の疎行列が高速化できた. 最も性能比が高い疎行列は 2.43 倍であり，パターンは 4×4，1 ブロックあたりの平均非零要素数の割合が 95%のデータ構造であった．

パターンごとにみると，1×4 では，BCRS 形式の効果は多くはなかった．4×1 では，性能比が 1〜2 倍のケースが多く, 平均で 1.09 倍となった．4×4 は，効果がある場合は 4×1 と近い性能がでている．一方，1 ブロックあたりの平均非零要素の割合が低い疎行列では，大きく性能が低下した．

表 1　各形式における最も性能が高い疎行列の数

1×1	1×4	4×1	4×4
349(26%)	8(0.6%)	881(66%)	95(7%)

各構造における最も性能が高い疎行列の個数を表 1 に示す．4×1 が 3 パターンの中で最も多く, 1333 種類中 881 種類の疎行列を高速化することができた．4×4 場合は 95 種類に留まった．

5. 自動チューニング

今回の実験では, 26%が CRS 形式のほうが良い．したがって，自動チューニングの考え方で最適な形式を選択するライブラリを試作した．この試作は反復解法ライブラリ Lis[5]をベースに行った．入力データとなる疎行列のデータ構造は一般的に用いられている MatrixMarket 形式とする．この入力データに対して CRS 形式と BCRS 形式の 3 つのパターンに変換し，それぞれ計測を行う．計測結果から最適な性能を得られる形式とその形式に変換した圧縮データを出力とした．最適形式に CRS 形式を加えることにより，CRS 形式のみの場合と比べて, 平均の倍率は 1.24 倍となった．

6. まとめ

AVX を用いて BCRS 形式の SpMV を実装した．BCRS 形式のブロックサイズとして, 1×1, 1×4, 4×4 の 3 パターンを用意した. BCRS 形式の SpMV の性能は CRS 形式の SpMV の性能と比較して，全体の 73%の疎行列で高速化できた．また，CRS 形式と 3 パターンの BCRS 形式から最適な形式を選ぶ自動チューニングライブラリを試作し，CRS 形式のみに比べ，平均の倍率は 1.24 倍となった．

参考文献

[1] R,Barrett., et al.: Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, SIAM, pp.57-58(1994)，長谷川里美，長谷川秀彦，藤野清次訳: 反復法 Templates, pp.77-79(1996)

[2] Intel: Intrinsics Guide，http: //software.intel.com/en-us/articles/intel-intrinsics-guide

[3] T.Hishinuma, A.Fujii, T.Tanaka, and H.Hasegawa: AVX acceleration of DD arithmetic between a sparse matrix and vector，PPAM2013

[4] The University of Florida Sparse Matrix Collection, http://www.cise.ufl.edu/research/sparse/matrices/

[5] 反復解法ライブラリ Lis，http://www.ssisc.org/lis/