AVX2を用いた倍精度疎行列と倍々精度ベクトルの積における精度と性能

菱沼 利彰[1], 藤井 昭宏[2], 田中 輝雄[2], 長谷川 秀彦[1]
[1]筑波大学, [2]工学院大学
e-mail: hishinuma@slis.tsukuba.ac.jp

1 概要

われわれは, 高速・安定な反復解法のために高精度演算の 1 つである倍々精度演算 [1]を SIMD 拡張命令 AVX2 (Advanced Vector Extensions 2)[2]を用いて高速化している [3].

AVX2 は, 1 命令で 4 つの倍精度演算を同時実行でき, FMA (Fused Multiply and Add)命令を使用できる.

反復解法ライブラリにおいて, 入力される疎行列Aは反復解法中で値を書き換えないことが想定されるため, 疎行列を倍精度, ベクトルを倍々精度とした混合精度疎行列ベクトル積 : $\boldsymbol{y}_{DD}=A_D\boldsymbol{x}_{DD}$ (DD-SpMV)を実装した.

本論文では, AVX2 を用いた CRS (Compressed Row Storage)形式 [4]と BCRS (Block CRS)形式 [4]の DD-SpMV の高速化の効果と, それらの実装が精度に与える影響について述べる.

2 AVX2 を用いた DD-SpMV

DD-SpMV の核である倍精度と倍々精度の積和演算は, FMA 命令を用いた場合, 19 回の倍精度演算で実装できる.

AVX2 を用いた CRS 形式の DD-SpMV は, 各行で A, $\boldsymbol{x}$の要素を 4 つずつレジスタに格納し, 積の結果を一時レジスタに足し込み, 各行の最後で一時レジスタ内の 4 つの要素を対応する $\boldsymbol{y}$の要素に足し込む.

これは, A に対応する $\boldsymbol{x}$の非連続なロード, $\boldsymbol{y}$への足し込みのための計算量の増加, 演算順序の変更などが必要となる. $\boldsymbol{y}$への足し込みのための倍々精度加算 3 回は, 倍精度加算 33 回からなり, 性能劣化要因となる.

一方, AVX2 に適した疎行列の格納形式として, BCRS 形式がある [3]. ブロックサイズを AVX2 の同時演算数に合わせた BCRS4x1 形式は, 疎行列を 0 要素を含む 4x1 の小密行列 (ブロック) の集合として格納する.

AVX2 を用いた BCRS4x1 形式の DD-SpMV は, CRS 形式と比べメモリアクセスを連続にし, $\boldsymbol{y}$への足し込みをなくせるが, ブロックに零要素を含むため計算量が最大で 4 倍増加する. 計算順序は, $\boldsymbol{y}$の要素に積の結果を逐次足し込む実装となるため, SIMD を用いない場合の CRS 形式の DD-SpMV と同じになる.

なお, AVX2 を用いた CRS 形式, BCRS 形式の DD-SpMV の実装方法は文献[3]で詳しく述べた.

3 数値実験

使用した CPU は intel Core i7 4770 4core 3.4GHz 16GB, OS は Fedora20, コンパイラは intel C/C++ Compiler 13.0.1, オプションは “-O3 -openmp -xCORE-AVX2 -fp-model precise” を用い, 実験はすべて OpenMP を用いた 4 スレッドで行った.

また, SIMD を用いない場合は“-xCORE-AVX2” のかわりに “-no-vec” を付けることでコンパイラによる自動 SIMD 化を抑制した.

対象とする問題は, The Univ. of Florida Sparse Matrix Collection [5]から正方, real, 非対称な問題のうち, N=10,000 以上で, 条件数の記載があるものを 12 問選んだ.

実験には, (1) AVX2 を用いない場合の CRS 形式の DD-SpMV (DD-Scalar), (2) AVX2 を用いた CRS 形式の DD-SpMV (DD-AVX2), (3) AVX2 を用いた BCRS4x1 形式の DD-SpMV (DD-BCRS)の 3 つの実装を用意した.

図 1 に, 各問題, 実装における実行時間と結果のベクトル $\boldsymbol{y}$の要素内で最大の相対誤差を示す. $\boldsymbol{x}$の値は-1 から 1 の範囲の倍精度の乱数とし, 相対誤差は, GMP ライブラリの多倍長浮動小数点数, 仮数部 1024bit を真値として求めた.

実験の結果, AVX2 や BCRS4x1 形式を用いた性能への影響として以下の結果を得た.

1) DD-AVX2 は DD-Scalar の約 4.5 倍から 8.0 倍高速.
2) DD-BCRS は DD-Scalar の 6.3 倍から 12.4 倍高速.
3) DD-BCRS は DD-AVX2 と比べて 0.95 倍から 1.83 倍高速. 今回の問題は BCRS4x1 形式にすることで, CRS 形式と比べて演算量が最大で 3 倍, 平均で 2.5 倍増加するが, ほとんどの問題に有効.

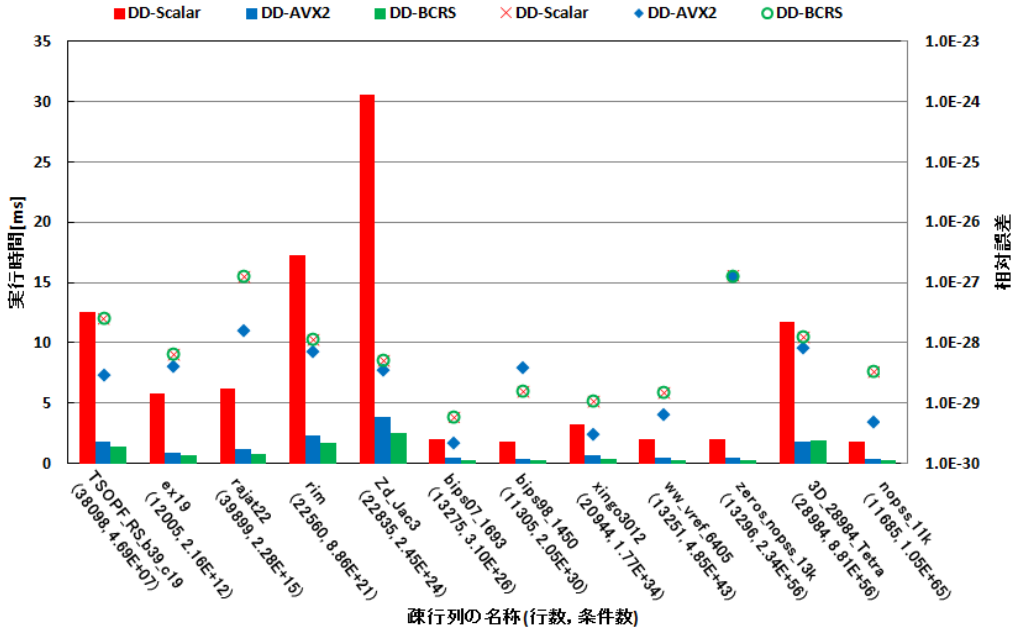

図 1 各実装における疎行列ベクトル積の実行時間 [ms]と相対誤差

また， AVX2 や BCRS4x1 形式を用いた DD-SpMV の計算精度について以下の 結果を得た．

1) DD-BCRS と DD-Scalar は計算順序が同じになるため，相対誤差は等しくなる．
2) AVX2 は他の実装と比べ計算結果を 4 つに分けて一時レジスタに格納し，各行の最後でレジスタ内の要素を y に足し込むため，相対誤差は最大で 1 桁程度小さい．
3) サイズや条件数による誤差の相関は得られなかった．

4 結論

本論文では，倍精度疎行列と倍々精度ベクトルの積を AVX2 や BCRS 形式を用いて高速化し，性能や計算精度への影響を調査した．

DD-AVX2 は DD-Scalar と比べ，4.5 から 8.0 倍高速である．また，DD-AVX2 は計算結果を 4 つに分けて一時レジスタに格納し，各行の最後でレジスタ内の要素を y に足し込むため，相対誤差が他の実装と比べて最大で 1 桁程度小さいことがわかった．

BCRS4x1 形式を用いた DD-BCRS は，メモリアクセスを連続にし，各行で発生する y への足し込みをなくせるが，ブロックに 0 要素を含むため計算量が増える．今回の問題では計算量が最大で 3.0 倍増加するが，DD-AVX2 と比べて 0.95 倍から 1.83 倍高速である．

BCRS4x1 形式は，演算順序が DD-Scalar と同じになるため，DD-AVX2 で得られた計算精度向上の効果は得られなくなるが，DD-Scalar と同様の計算精度は得られることがわかった．

今後は，実際に反復解法に対するこれらの実装の影響について，評価する必要がある．

参考文献

[1] Bailey, D ,H.: High-Precision Floating-Point Arithmetic in Scientific Computation, computing in Science and Engineering (2005), pp. 54-61.
[2] Intel: Intrinsics Guide, http://software.intel.com/en-us/articles/intel-intrinsics-guide/.
[3] 菱沼 利彰，藤井 昭宏，田中 輝雄，長谷川 秀彦, AVX2 を用いた倍精度 BCRS 形式疎行列と倍々精度ベクトル積の高速化，情報処理学会論文誌 コンピューティングシステム，Vol.7，No.4 (2014)，pp.1-10.
[4] Barrett, R., et al.: Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, SIAM, 1994, pp. 57-65.
[5] The University of Florida Sparse Matrix Collection, http://www.cise.ufl.edu/research/sparse/matrices/.