1J-06

倍々精度 SpMV における SIMD 利用時のデータレイアウトによる性能分析

慈道 亮人† 寺田 洋人† 菱沼 利彰‡ 藤井 昭宏† 田中 輝雄†
工学院大学† ブレイドテクノロジーズ株式会社‡

1. はじめに

高精度演算のひとつに倍々精度 (DD) 演算 [1] がある．これは 2 つの倍精度数で 1 つの数を表し，倍精度演算を 10〜20 回行って 4 倍精度相当の演算を行う．

DD 型ベクトルのデータレイアウトに，AoS (Array of Structures) と SoA (Structure of Arrays) がある．SIMD (Single Instruction, Multiple Data) 命令は複数のデータをまとめて処理するため，性能を出すにはデータレイアウトを慎重に選択する必要がある [2].

疎行列とは成分の多くが零の行列である．零成分を圧縮することでデータ量を削減できる．疎行列の格納形式に，CSR (Compressed Sparse Row) 形式と ELL (ELLPACK) 形式がある [3].

福永らによって，SIMD 環境下での DD 型ベクトルのデータレイアウトが，性能に与える影響は調べられている [4]．演算対象は，ベクトル演算 axpy ($\boldsymbol{y} = \alpha\boldsymbol{x} + \boldsymbol{y}$) と dot ($\langle \boldsymbol{x}, \boldsymbol{y} \rangle = \boldsymbol{x}^\top \boldsymbol{y}$)，CSR 形式疎行列ベクトル積 SpMV ($\boldsymbol{y} = A\boldsymbol{x}$) である．本論文では，SpMV は倍精度疎行列と DD 型ベクトルの積とする．

本論文では SIMD 利用時の倍々精度 SpMV の性能に，データレイアウトと格納形式が与える影響を分析する．

2. データレイアウト

複数の値からなるデータを，多値データと呼ぶ．多値データのデータレイアウトである AoS と SoA は，メモリアクセスパターンが異なるため，性能に影響を及ぼす [5]．DD 型のメンバを “hi”, “lo” として，AoS と SoA のデータレイアウトを図 1 に示す．

AoS では，メンバが交互に並ぶ．そのため，SIMD では非連続メモリ操作が必要になる．標準的なデータレイアウトであり，多くのライブラリでサポートされている．

SoA では，各メンバが連続して並ぶ．そのため，SIMD による高速化に適している．

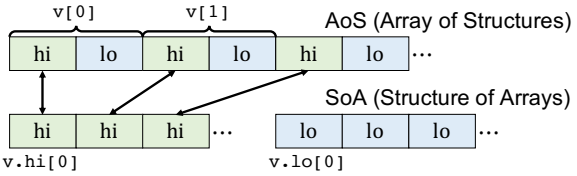

図 1 DD 型ベクトルのデータレイアウト

3. 疎行列格納形式

本論文では CSR 形式に加えて，ELL 形式を用いて評価を行う．

CSR は，非零成分を左に詰め，1 行目から順に 1 つの配列に格納する．行方向に連続なため，ベクトル $\boldsymbol{y}$ の各要素へのアクセスは 1 回のみになる．

ELL は，非零成分を左に詰め，1 列目から順に 1 つの配列に格納する．非零成分がない場合は，零とする．列方向に連続なため，ベクトル $\boldsymbol{y}$ 全体へのアクセスが複数回必要になる．そのためキャッシュ効率が悪い．これを改善するためにブロッキングを行う．本論文ではブロッキングを行なった場合を報告する．

4. SIMD 化の方針

Intel の SIMD 拡張命令 AVX2，AVX512 を使用する．倍精度数での，それぞれの並列度は 4, 8 である．表 1 に，最内側ループでの，各ベクトルの要素へのアクセスと，その際のメモリ操作の連続性を示す．

ベクトル $\boldsymbol{x}$ は，CSR, ELL ともに，set, gather の二つの方法でロードした．set は，複数の指定した値を，ロードする方法である．直接対応する命令はなく，コンパイラによって命令列が生成される．gather は，ひとつのベースアドレスと複数のインデックスを指定して，ロードする方法である．

ベクトル $\boldsymbol{y}$ のロードには，CSR ではスカラ load 命令を使用した．ELL では，SoA の場合はベクトル load 命令，AoS の場合は set, gather, permute, shuffle, unpack の五つの方法でロードした．permute, shuffle, unpack はレジスタ間で値を入れ替える方法である．

ここでは AVX2，gather の場合を報告する．

表 1 各ベクトルのアクセスとメモリ操作

		ベクトル $\boldsymbol{x}$		ベクトル $\boldsymbol{y}$	
		アクセス	メモリ操作	アクセス	メモリ操作
CSR	AoS SoA	非連続	非連続	スカラ	スカラ
ELL	AoS SoA	非連続	非連続	連続	非連続 連続

Performance analysis of data layout for SIMD accelerated Double-Double precision SpMV
Ryoto Jido† Hiroto Terada† Toshiaki Hishinuma‡ Akihiro Fujii† Teruo Tanaka†
† Kogakuin University
‡ Braid Technologies K.K.

表 2　実験環境と計測条件

実行環境	Oakbridge-CX
CPU	Intel Xeon Platinum 8280 × 2
コア数	56
L1D キャッシュ	32 KiB/core
L2 キャッシュ	1 MiB/core
L3 キャッシュ	1.375 MiB/core　(28 コアで共有)
メモリバンド幅	281 GB/s
コンパイラ	Intel C++ Compiler 19.1.3
	-O3 -std=c++17 -fp-model precise -fma
	-xCORE-AVX2 -qopenmp
計測条件	1 ノード，56 スレッド
	計測の前に 1 回実行
実行時間	100 回実行する時間を，100 で割った値

表 3　使用した実問題疎行列

name	row	nnz	一行の最大 nnz
af_shell1	504,855	17,588,875	40
ecology1	1,000,000	4,996,000	5
nlpkkt160	8,345,600	229,518,112	28

5.　実験

5.1　実験概要

OpenMP によるマルチスレッド化を行なった．ブロッキングサイズは，ひとつのブロックの処理に使うデータ量が，1 コアの L2 キャッシュに収まるサイズにした．そのためブロッキングサイズは，疎行列ごとに異なる．表 2 に実験環境と計測条件を示す．

5.2　結果と分析

対象として，実問題疎行列と帯行列を計測した．実問題疎行列は，SuiteSparse Matrix Collection から入手した，表 3 に示す行列を使用した．帯行列は，行数を 10^5 に固定し，帯幅を 1 から 100 まで変化させた．

5.2.1　実問題疎行列

結果を図 2 に示す．縦軸は性能で，1 秒間に行なった倍精度演算の回数 (GDFLOPS) である．

CSR の場合，AoS と SoA の性能差は少なかった．これはベクトル $\boldsymbol{x}$ のメモリ操作が，AoS, SoA ともに非連続だからである．ELL の場合，SoA のほうが AoS より性能が最大 1.5 倍良かった．これはベクトル $\boldsymbol{y}$ のメモリ操作が，SoA では連続，AoS では非連続だからである．ベクトル $\boldsymbol{x}$ のメモリ操作は，AoS, SoA ともに非連続である．

5.2.2　帯行列

図 3 のグラフの，縦軸は性能，横軸は帯幅である．

CSR では，各帯幅で AoS と SoA の性能差は少ない．また SIMD 化に関わる端数処理が発生しないため，帯幅が 4 の倍数ごとに性能が向上している．ELL では，全ての帯幅で SoA の方が性能が高く，最大 1.3 倍の差があった．またブロッキングによるブロック数が増える帯幅で，性能が低下している．

CSR と ELL の SoA を比較すると，帯幅 30 以下では ELL の方が性能が良く，帯幅 30 より大きいと性能差は少ない．これは連続方向の違いにより，CSR では帯幅が小さいとき SIMD 処理される要素が少ないが，ELL では帯幅に関わらず多くの要素が SIMD 処理されるからである．

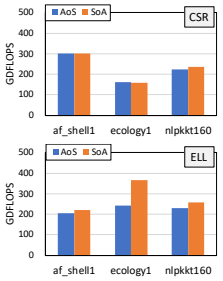

図 2　性能（実問題）

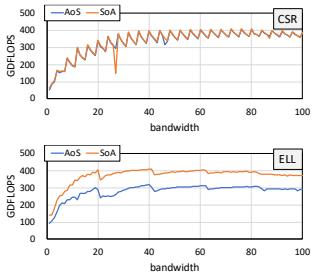

図 3　性能（帯行列）

6.　おわりに

本論文では，DD 型ベクトルと倍精度疎行列の SpMV の性能に，データレイアウト（AoS, SoA）と格納形式（CSR, ELL）が与える影響を調べた．結果を次にまとめる．

- CSR では，全て非連続なメモリ操作なので，データレイアウトによる性能差は少なかった．
- ELL では，ベクトル $\boldsymbol{y}$ が連続なメモリ操作になる，SoA の方が性能が良かった．
- 帯行列において，CSR が SIMD の加速を受けづらいため，帯幅 30 以下では ELL の方が性能が高く，それ以上では差はなかった．

これらより，CSR ではライブラリのサポートが多い AoS を使用し，ELL では性能面で有利な SoA を使用すると良い．また 1 行当たりの非零要素数が 30 以下の場合は性能面で有利な ELL を，30 より大きい場合はメモリ量で有利な CSR を，採用すると良い．

今後の課題として，他の環境での計測，その他の格納形式での計測が挙げられる．

謝辞

本研究は，JSPS 科研費 JP18K11340 の助成により実施した．

参考文献

[1] Bailey, D.H.: High-precision floating-point arithmetic in scientific computation, Computing in Science & Engineering, Vol.7, No.3, pp.54-61 (2005).

[2] Watanabe, H., Nakagawa, K.M.: SIMD vectorization for the Lennard-Jones potential with AVX2 and AVX-512 instructions, *Comp. Phys. Comm.*, Vol.237, pp.1-7 (2019).

[3] Yousef, S.: Iterative Methods for Sparse Linear Systems. pp.92-95, Society for Industrial and Applied Mathematics (2003).

[4] 福永晋司，山浦朴人，菱沼利彰，藤井昭宏，田中輝雄：倍々精度演算における SIMD 命令利用時のデータレイアウトによる性能差分析，情報処理学会 第 84 回全国大会，4J-03（2022）．

[5] Wen-mei, W.H. (Eds.): GPU Computing Gems Jade Edition, Strzodka, R.: *Abstraction for AoS and SoA layout in C++*, pp.429-441, Elsevier (2012).