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

福永 晋司[†]　山浦 朴人[†]　菱沼 利彰[‡]　藤井 昭宏[†]　田中 輝雄[†]

工学院大学[†]　株式会社科学計算総合研究所[‡]

1. はじめに

数値シミュレーションの核である反復法は丸め誤差が収束に悪影響を及ぼす [1]．丸め誤差による影響を改善する方法に高精度化があり，計算時間やデータ量は増加するが収束の改善に有効であることが確認されている [2].

高精度演算のひとつに倍々精度 (DD)演算 [3]がある．Bailey らが提案した DD 演算は倍精度型変数を 2 つ組合せて擬似的に 4 倍精度演算を行う手法である．DD 演算では倍精度演算と比較して 10〜20 倍の演算量を必要とする．

DD 型の配列として考えられる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) [4]では複数のデータをまとめて処理するため，効率よく利用するためにはデータレイアウトの選択に注意する必要がある [5].

また近年の SIMD 環境では DD 演算におけるデータレイアウト同士の比較については確認できず，データレイアウトの違いが性能に与える影響は明らかにされていない．

本論文の目的は DD 演算における SIMD 利用時の AoS と SoA の特徴を明らかにすることである．

2. データレイアウト

データレイアウトは多値データを扱う際のメモリ上のデータ構造であり，メモリアクセスパターンが変わるため性能に影響する [6]．DD 型の上位のデータを“hi”，下位のデータを“lo”とした AoS と SoA を図 1 に示す.

AoS は hi と lo が交互に配置されるデータレイアウトである．標準的なデータレイアウトであり，多くのプログラミング言語やライブラリでサポートされている．一方で SIMD の利用には非連続なメモリ操作が必要となる.

SoA は hi と lo がそれぞれ連続して配置されるデータレイアウトである．SIMD の利用において連続したデータを同時に処理することができるため SIMD を用いた高速化に適している．欠点はサポートされているライブラリが少ないため利用には工夫が必要となることである.

3. SIMD 化の方針

SoA は load 命令 / store 命令を用いてメモリの連続アドレスからデータをロード / ストアできる．一方，AoS は非連続なメモリ操作である gather 命令 / scatter 命令や複数の SIMD 命令の組合せで実現する必要がある.

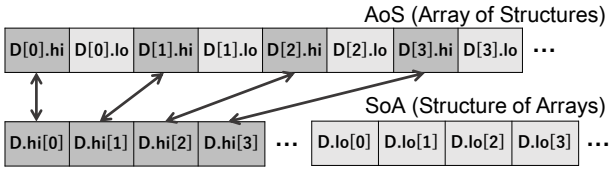

図 1　DD 型の AoS と SoA

本論文では AoS のデータのロードに関していくつかの方法を試行した．ベクトル演算ではそれぞれ主に (a) set 命令，(b) gather 命令，(c) shuffle 命令，(d) unpack 命令，(e) permute 命令を利用した計 5 パターンを実装した．SpMV では DD ベクトル $\boldsymbol{x}$ へのアクセスが非連続であるため，AoS と SoA のどちらも set 命令を利用した.

4. 演算ごとの性能比較

4.1 実験概要

評価指標としてベクトル演算 axpy ($\boldsymbol{y} = \alpha\boldsymbol{x} + \boldsymbol{y}$)と dot ($\langle \boldsymbol{x}, \boldsymbol{y} \rangle = \boldsymbol{x}^{\mathrm{T}}\boldsymbol{y}$)，疎行列ベクトル積 SpMV ($\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$)を用いた評価を行った．α は DD スカラ，$\boldsymbol{x}$ と $\boldsymbol{y}$ は DD ベクトル，$\boldsymbol{A}$ は倍精度疎行列であり，疎行列格納形式には CRS (Compressed Row Storage)形式を用いた．また各演算に OpenMP を用いたマルチスレッド化と SIMD 化を行った.

実験には東京大学の大規模超並列スーパーコンピュータ“Oakbridge-CX” [7]を使用した．1 ノードあたり CPU Intel Xeon Platinum 8280 が 2 基搭載され，コア数 56 コア，L3 キャッシュ 77 MB，メモリバンド幅 281.6 GB/s である．コンパイラは Intel C/C++ compiler 18.0.2 を用いた．以降，キャッシュは L3 キャッシュを指す.

計測方法については以下の通りである.

- 1 ノード，56 スレッド（1 スレッド/コア）
- 約 1 秒間実行，1 回の演算時間には $\frac{\text{実行時間}}{\text{実行回数}}$ を使用
- 計測前に 1 回実行した状態で計測（ホットスタート）
- コンパイルオプション：-std=c++17 -xHost -fp-model precise -fma -O3 -parallel -qopenmp -mkl=parallel (-axcore-avx2 または -axcore-avx512)
- 実行時コマンド：numactl --localalloc

表 1　実問題疎行列

#	name	rows	nnz	nnz/row
A	thermal2	1,228,045	8,580,313	6.9
B	parabolic_fem	525,825	3,674,625	6.9
C	bmwcra_1	148,770	10,641,602	71.5
D	x104	108,384	8,713,602	80.3
E	crankseg_1	52,804	10,614,210	201.0
F	nd3k	9,000	3,279,690	364.4

Performance Analysis of Data Layout
of SIMD Accelerated DD Precision Operations

Shinji Fukunaga[†], Naoto Yamaura[†],
Toshiaki Hishinuma[‡], Akihiro Fujii[†], Teruo Tanaka[†]
† Kogakuin University
‡ Research Institute for Computational Science Co. Ltd.

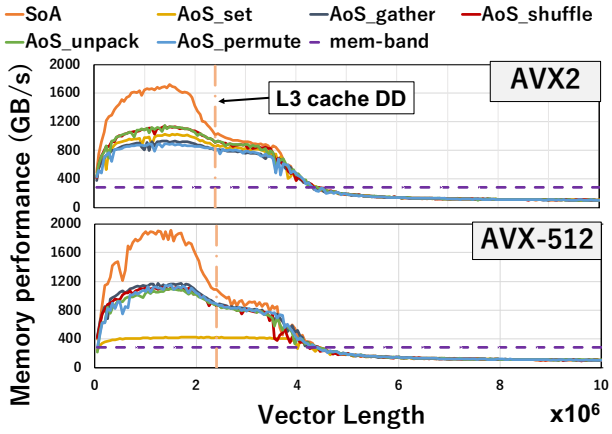

図 2　axpy メモリ性能

実験 1　axpy

ベクトル長を変化させた axpy の結果を図 2 に示す．縦軸はメモリ性能 (GB/s)であり，L3 cache DD は DD ベクトル 2 本がキャッシュに収まるベクトル長である．

まず AoS の 5 パターンのロード方法に注目すると，AVX2 では (c) shuffle 命令と (d) unpack 命令，AVX-512 では (b) gather 命令を用いたパターンの性能が最も高かった．AVX-512 では (a) set 命令を用いたパターンの性能が他のパターンに比べて大きく低下しているのに対して AVX2 では同程度の性能が出ている．ここで set 命令は他のパターンの命令とは異なりコンパイラの判断により一連の命令が生成されることに注意すると，AVX-512 では最適な命令列の生成が行われていない可能性がある．

次にベクトル長の変化に注目する．ベクトル長がキャッシュに収まる場合，SoA は AoS に対して AVX2 では最大約 1.6 倍，AVX-512 では最大約 1.7 倍の性能が出た．これは AoS ではデータのロードのために非連続なメモリ操作や複数の SIMD 命令を実行しているためである．キャッシュに収まらないベクトル長ではメモリ性能に制限されて SoA と AoS の性能差はなくなった．

4.2　実験 2　SpMV（実問題疎行列）

実問題疎行列に対する SpMV の性能を分析するために，SuiteSparse Matrix Collection から入手した 6 種の対称正定値行列を用いる．計測に用いた疎行列を表 1 に示す．

実問題疎行列に対する SpMV の結果を図 4 に示す．縦軸は 1 秒間に実行した倍精度演算の回数を用いた性能(GDFLOPS)であり，横軸は表 1 の疎行列に対応する．6 種の疎行列すべてにおいて AoS と SoA の性能差は小さかった．これは SpMV ではデータレイアウトによらず非連続なメモリ操作が必要になるためである．また AVX2 と比較して AVX-512 の性能が半分程度である原因のひとつとして，動作クロック周波数の低下や set 命令の最適な命令列の生成が行われていないことが挙げられる．

4.3　実験 3　SpMV（帯行列）

平均非零要素数が性能に影響するかを分析するために，1 行あたり m 個の非零要素が連続する帯行列を用いる．帯行列の行数は 10^5 で固定し，帯幅 m を変化させた結果を図 4 に示す．縦軸は性能 (GDFLOPS)である．

AoS と SoA の性能差は各帯幅で小さく，平均非零要素数がメモリレイアウトの違いに与える影響は小さい．また帯幅の増加に伴って性能が向上している．これは SIMD 化によって高速化される要素の割合が増加するためである．さらに SIMD 化に係る端数処理を必要としない 4 の倍数や 8 の倍数の帯幅では性能が上がっている．

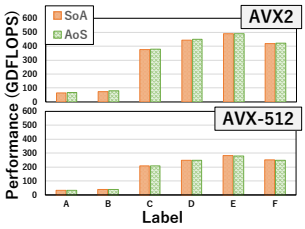

図 4　SpMV（実問題）

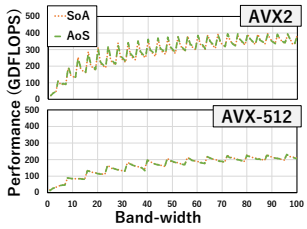

図 4　SpMV（帯行列）

5. おわりに

本論文では SIMD を利用したベクトル演算と SpMV を実装し，DD 演算におけるデータレイアウト分析を行った．

axpy ではキャッシュに収まる場合，SoA は AoS より良い性能であった．一方でキャッシュに収まらない場合，性能はメモリ性能に制限されてデータレイアウトは性能に影響しなかった．また SpMV ではデータレイアウトによらず非連続なメモリアクセスが必要となるため SIMD 化のコストによる性能差は生じなかった．

DD 演算では倍精度演算に比べて演算強度 (Flop / Byte) が増加するため性能がボトルネックになりやすいが，近年のコンピュータでは CPU 性能に対してメモリ性能が劣るためメモリ性能がボトルネックになりやすい．

このことから扱う問題や目的に応じてデータレイアウトを選択することが想定される．例えば問題サイズが小さくキャッシュを有効に活用できる場合は SoA を用いた高速化の利点を得られる．他方，問題サイズが大きくキャッシュを活用できない場合やメモリ性能がボトルネックとなる演算が多くを占める場合にはライブラリで広くサポートされている AoS の方が利便性が高い．

今後の課題として性能がボトルネックとなる演算やアーキテクチャの異なる環境での評価が挙げられる．

謝辞

本研究は JHPCN 拠点の支援（課題番号:jh210019-NAH）及び JSPS 科研費 JP18K11340 の助成により実施した．

参考文献

[1] Greenbaum, A.: *Iterative methods for solving linear systems*, pp.61–75, SIAM (1997).

[2] Hasegawa, H.: Utilizing the quadruple-precision floating-point arithmetic operation for the Krylov Subspace Methods, Proc. *The 8th SIAM Conference on Applied Linear Algebra*, Citeseer (2003).

[3] Bailey, D.H.: High-precision floating-point arithmetic in scientific computation, Computing in science & engineering, Vol.7, No.3, pp.54–61, DOI: 10.1109/MCSE.2005.52 (2005).

[4] Intel Corporation: Intel Intrinsics Guide, Intel Corporation (online), available from 〈https://software.intel.com/sites/landingpage/IntrinsicsGuide/〉 (accessed 2022-01-07).

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

[6] 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).

[7] 東京大学情報基盤センター　スーパーコンピュータ部門：Oakbridge-CX スーパーコンピュータシステム，東京大学情報基盤センター　スーパーコンピュータ部門（オンライン），入手先〈https://www.cc.u-tokyo.ac.jp/supercomputer/obcx/〉（参照 2022-01-06）．