Knights Landing における倍々精度基本演算のハイブリッド並列の特性評価

伊藤友太† 土肥樹† 菱沼利彰‡ 藤井昭宏† 田中輝雄†

工学院大学† 筑波大学‡

1 はじめに

我々は，倍々精度演算の高速化の研究をすすめている．倍々精度演算は，倍精度演算と比較して 2 倍のメモリリソースと 10 倍以上の演算量が必要になる [1]．一方で，Knights Landing（KNL）というメニーコアプロセッサが様々なシステムで利用されている．メニーコアプロセッサは，演算性能とメモリ性能の関係からメモリのバンド幅に性能が律速されやすい[2]．しかし，倍々精度演算においてコストがかかっていた演算量を隠匿できる環境と言い換えられる．

これらを踏まえて本稿では，アフィニティとスレッド分割数，KNL の AVX-512 によるチューニングの有効性， OpenMP と MPI によるハイブリッド並列について倍精度と倍々精度のベクトル加算を用いて実験と考察を行う．

2 倍々精度演算

倍々精度は Bailey[3]が提案した倍精度データを 2 つ組み合わせ 4 倍の精度を保持する方法である．倍々精度の四則演算は丸め誤差を保持する倍精度加算と乗算のアルゴリズムに基づく．

本稿では倍々精度ベクトル加算 (DDV)，倍精度ベクトル加算（DV）を実装し考察する．DV はベクトルの 1 要素あたり，1 回の演算，24Byte のメモリリソースが，DDV は 11 回の演算，48Byte のメモリリソースが必要になる．1 秒あたりの倍々精度演算回数を DD-FLOPS と定義する．

3 KNL

図 1は KNL のブロックダイアグラムである．KNL にはタイルと呼ばれるセットがある．タイルは 2 つのコア，それらが共有する 1MB の L2 キャッシュ，コアごとに 2 つのベクトル処理ユニット（VPU）で構成される．そのタイルを

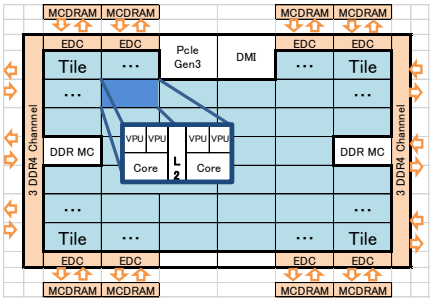

図 1 KNL ブロックダイアグラム

36 個並べ 2 次元メッシュで相互に接続する．MCDRM（2GB×8，450GB/s）のメモリをもつ．命令セットは AVX-512 が追加され，8 個の倍精度積和演算を 1 つの SIMD 命令で実行できる．

4 実験と考察

4.1 実験環境

インテル® Xeon Phi™ プロセッサー 7250 を使用した．アクティブなタイルは 34 個のためコアは 68 個，コア当りのスレッド数は 4 つのため最大 272 スレッドとなる．今回の実験では OS の監視に使用されるタイルを 1 つ以上外す．コンパイラは icc バージョン 17.0.1 を使用，オプションは -O3， -xMIC-AVX512，-qopenmp を使用した．実行時オプションに numactl –m 1 を付与することでデータは MCDRAM のみに割り当てるように設定した．また，AVX-512 で DV，DDV の SIMD 化を行った．

4.2 アフィニティとスレッド分割数

KNL の前世代では，コア当りに 2 スレッド以上立てないとリソースを使いきれなかった．これに対して KNL はコア当り 1 つのスレッドで全てのリソースを有効に使えると言われている[2].

スレッド分割数の検証を行った．OpenMP でスレッド並列度を変化させた．アフィニティは，なるべく使用するコアを少なくする compact と，多くする scatter の 2 つのパターンとし DV，DDV を計測した．ベクトルサイズは L2 キャッシュに収まらない 2^{23} とした．OS の監視の影響を避けるために 32 タイル，64 コアを使用した．結果を図 2に示す．

Evaluation of Hybrid Parallelism of DD Arithmetic Basic Operation on Knights Landing

Yuta Ito†, Tatsuki Doi†, Toshiaki Hisinuma‡, Akihiro Fuhii† and Teruo Tanaka†

Kogakuin University†, University of Tsukuba‡

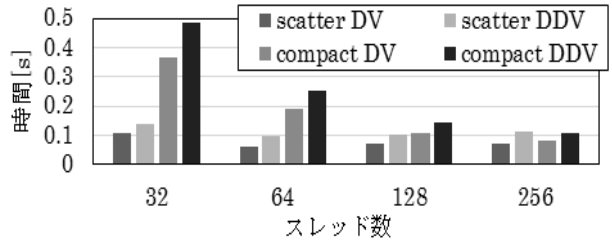

図 2 スレッド数とアフィニティ

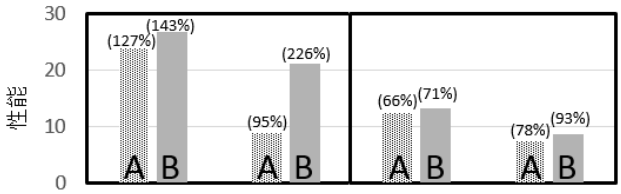

図 3 SIMD 化 (括弧は MCDRAM 転送率との比較)
上部：ベクトルサイズ=2^{19}　下部：ベクトルサイズ=2^{23}

どちらの精度でも 64 スレッドのアフィニティを scatter とした場合が一番良い性能だった．
1 コアに 1 スレッドを立てる設定の方が 4 スレッドを立てるより性能が高くなること示している．また，DV，DDV どちらも同じ傾向のため，演算量は影響しないと考えられる．

4.3 AVX-512 によるチューニングの有効性

AVX-512 命令の有効性の分析を行うために DV，DDV を，AVX-512 命令の利用の有（A）と無（B）で比較した．A にはコンパイルオプション- no-vec を追加した．スレッド数は 64，アフィニティは scatter，ベクトルサイズは L2 キャッシュに収まる 2^{19} と収まらない 2^{23} の 2 種類とした．演算を複数回反復し平均を取った．結果を図 3 に示す．

全てのパターンにおいて A より B の方が良い結果となった．その中でも特に L2 キャッシュに収まるベクトルサイズの DDV は AVX-512 によるチューニングの影響が大きく，A より B が 2.38 倍良い結果となった．これは SIMD 化による演算性能の増加によるためと考えられる．また，AVX-512 を用いることでデータ供給能力が向上されることも要因のひとつである．一方で，ベクトルサイズが大きく L2 キャッシュに収まらない場合，DDV は DV よりも 2 倍のロードストア命令が必要になるため本来は性能が半分になる．しかし実際には MCDRAM のデータ転送性能に律速されるため A は約 6 割，B は約 6.5 割の性能を保ち MCDRAM の帯域幅の 93%を使用できた．

以上のことから，AVX-512 はキャッシュに収まるか否かは関係なく有効であり，演算強度が低いほど影響が大きい．

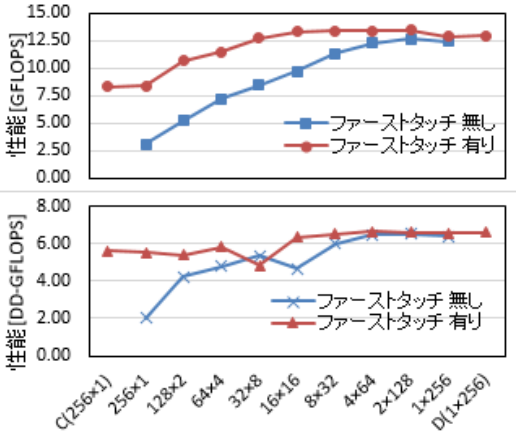

図 4 DV，DDV のハイブリッド並列性能
上部：DV　下部：DDV

4.4 OpenMP と MPI のハイブリッド並列

スレッド並列時に考慮するファーストタッチの影響と，MPI と OpenMP のハイブリッド並列の効果を調べた．分析には DV，DDV のハイブリッド並列を実装し，スレッドとプロセスの比率を変化させながら計測した．ベクトルサイズはキャッシュに収まらない 2^{23} として，演算を複数回反復し平均を取った．さらにファーストタッチの有無とスレッド並列のみ（C），プロセス並列のみ（D）を計測した．結果を図 4に示す．

ファーストタッチは DV，DDV ともに，ほとんどの比率で機能した．スレッド並列よりもハイブリッド並列や MPI によるプロセス並列のみ方が良い結果が得られた．

5 おわりに

4.2節の実験から単純な演算であればコアごとに 1 スレッドを割り当てた方が良い性能が得られた．4.3節の実験から AVX-512 によるチューニングの有効性が得られた．4.4節の検証からスレッドの並列化を行う際はスレッドのファーストタッチに配慮する必要があること，スレッド並列よりもハイブリッド並列やプロセス並列のみ方が良い性能が引き出せることが分かった．

今後の課題としてキャッシュモードなどの KNL に搭載されているが今回考慮していない部分の影響度についてさらに追及していきたい．

参考文献

[1] 菱沼，藤井，田中，長谷川，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 ACS vol.7 no.4，pp.25-33，2014

[2] J Jeffers, J Reinders, A Sodani, Intel® Xeon Phi™ Processor High Performance Programming Knights Landing Edition, Morgan Kaufmann, 2016

[3] Bailey, High-Precision Floating-Point Arithmetic in Scientific Computation, Computing in Science and Engineering, pp.54-61, 2005