2.5 次元アルゴリズムを用いた高性能 PDGEMM の開発

椋木大地
今村俊幸
理化学研究所 計算科学研究センター

１．はじめに

本稿では 2017 年度 Oakforest-PACS スーパーコンピュータシステム「大規模 HPC チャレンジ」の第 2 回課題に採択された「2.5 次元アルゴリズムを用いた高性能 PDGEMM の開発」（2018 年 1 月 25 日～26 日に実施）の結果について報告する。

スーパーコンピュータの性能向上はノードあたりの性能向上に加えノード数の増加によって成し遂げられている。一方で演算性能の進化に対してネットワーク性能の進化は緩やかであることや，並列数が増加するにつれて集団通信における通信レイテンシがボトルネックとなり性能が悪化することから，大規模並列計算においてアプリケーション性能が通信コストに律速される傾向が年々高まっており，超並列環境を想定した通信削減手法の研究や通信削減型アルゴリズムをサポートした数値計算ライブラリの開発が急務となっている。

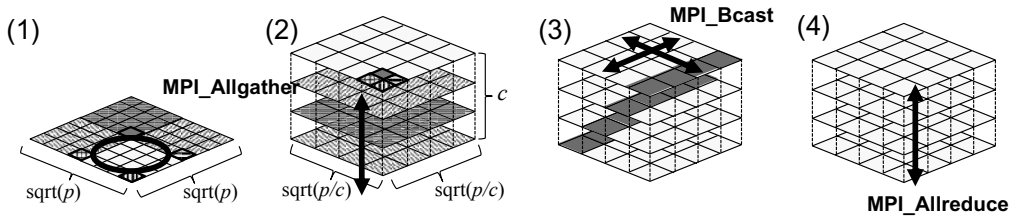
このような問題は，一般に性能が演算律速となる処理とみなされる行列積においても起こりうる。行列積は $O(n^2)$のデータ参照に対して $O(n^3)$の演算があり，単体ノードでの計算や並列計算において 1 ノードあたりの計算サイズが十分に大きい場合には，性能は計算機の演算性能に律速する。ところが強スケーリング性能を考えた時に，1 ノードあたりの計算サイズが十分に確保できなくなると，やがて通信コストが性能律速要因となることが知られている。そのため行列積の並列計算に対しても通信削減型のアルゴリズムである 2.5 次元アルゴリズムが提案されている[1]。

ScaLAPACK（PBLAS）で提供されている倍精度並列行列積ルーチン（PDGEMM）は科学技術計算の基本演算カーネルの一つであり，我々は次世代の超並列計算機環境に向けて，強スケーリングの観点で高い性能を持った PDGEMM を提供するために,2.5 次元アルゴリズムを用いた PDGEMM（2.5D-PDGEMM）の開発を進めている。2.5 次元アルゴリズムは 3 次元のプロセスグリッド上に 2 次元分散された行列をスタックして計算するため，ScaLAPACK で提供されている PDGEMM 等の 2 次元アルゴリズムによる実装（2D-PDGEMM）の代用とするには，行列データの再分散が必要となる。2.5D-PDGEMM の実装・評価はこれまでに行われていたものの，我々の知る限り行列再分散の問題をクリアし従来の PDGEMM と互換性のある実装を行った例は報告されていなかった。そこで我々は理論プロセスグリッドの再配置と行列再分散を行うことにより，2D-PDGEMM と互換性を確保した 2D 互換 2.5D-PDGEMM を実装し，これまでにスーパーコンピュータ「京」において性能評価を行ってきた[2]。しかし京は運用開始から 6 年以上が経過し，近年の計算機環境とは構成・性能特性が大きく異なる環境となっている。そこで本研究では世界最高クラスの性能を誇る Oakforest-PACS[1]を活用することで，より現代的な大規模計算環境における性能を分析することとした。

[1] 本研究の測定実施時点で最新の 2017 年 11 月付け Top500 リスト（https://www.top500.org/）においては世界 9 位のシステムであった

２．2.5 次元アルゴリズムの概要と 2 次元互換実装

従来から広く用いられている Cannon や SUMMA 等の並列行列積アルゴリズム（2 次元アルゴリズム）は，2 次元プロセスグリッド上に 2 次元分散（2 次元ブロックあるいは 2 次元ブロックサイクリック分散）された行列を，部分行列の交換と計算を繰り返しながら複数ステップを経て計算を完了する。一方の 2.5 次元アルゴリズムは，3 次元プロセスグリッド（x×y×z）を想定し，その各 2 次元平面（x-y）上に 2 次元分散された行列が垂直方向（z 方向）に複製されて積み重ねられた（スタックされた）状態で計算を行う（この分散状態を本稿では 2.5 次元分散と呼ぶ）。したがって，2.5D-PDGEMM に 2D-PDGEMM と互換性を持たせるには，2D-PDGEMM が想定する 2 次元プロセスグリッドから論理的な 3 次元プロセスグリッドを作るとともに，そこに行列を 2.5 次元分散しなければならない。2.5 次元アルゴリズムは，垂直方向に複製された行列の冗長性を利用することで，2 次元アルゴリズムの計算ステップを垂直方向の行列スタック数で分割して並列に計算することにより，計算ステップ数および通信回数を削減する。すなわち，まず 3 次元プロセスグリッド上の各 2 次元平面上で，2 次元アルゴリズムの部分計算ステップを実行することにより行列積の一部分を計算する。そして最後に各 2 次元平面上で得られた計算結果を垂直方向にリダクションすることで，行列積の計算が完了する。総プロセス数を p とするとき，2 次元プロセスグリッド $p^{1/2}\times p^{1/2}$ に 2 次元分散されている n×n の行列を 2 次元アルゴリズムで計算する場合の必要な通信回数は $O(p^{1/2})$ である。一方，2.5 次元アルゴリズムによって行列スタック数 c として 3 次元プロセスグリッド $(p/c)^{1/2}\times(p/c)^{1/2}\times c$（ただし $c\leq p^{1/3}$）を構成して計算を行う場合，通信回数は $O(p^{1/2}/c^{3/2})$ となる。このとき演算コストは $O(n^3/p)$ で変化しないが，メモリ量は $O(n^2/p)$ から $O(cn^2/p)$ に増加する（ワーク配列は除く）。したがって，メモリ量と通信回数のトレードオフが存在する。



第 1 図：実装の概要

図 1 に我々の 2D 互換 2.5D-PDGEMM の実装を模式的に示す。まず図 1 (1)は総プロセス数が p で，2 次元プロセスグリッド $p^{1/2}\times p^{1/2}$ において行列が 2 次元分散されている状態を示している。2D-PDGEMM はこの状態で $p^{1/2}$ ステップの行列交換（通信）と演算を経て行列積を完成させる。一方，我々の実装はこの状態から，行列を行列スタック数で 2 次元ブロック分散のように等分し（図の例はスタック数 c=4 の場合を示している），MPI_Comm_split を用いて図 1 (2)のような論理的な 3 次元プロセスグリッドを切り出す。ここでは 3 次元プロセスグリッドの各 2 次元平面に相当する水平方向通信用の MPI サブコミュニケータを作成するとともに，垂直方向通信用の MPI サブコミュニケータも作成する。垂直方向通信用の MPI サブコミュニケータ上で MPI_Allgather を実行することにより，2.5 次元分散への再分散が完了する。続いて水平方向通信用の MPI サブコミュニケータ上で 2 次元アルゴリズムの 1/c ステップを実行して部分的な行

列積の計算を行う（図 1(3)）。ここには任意の 2 次元アルゴリズムを利用できるが，本研究では MPI_Bcast のみで通信を構成できる SUMMA[3]を用いた。計算は各ノード上で Level-3 BLAS ルーチンの DGEMM によって行われる。最後に各層で計算された部分的な行列積の結果を，垂直方向通信用の MPI サブコミュニケータ上でリダクションすることにより計算が完了する（図 1(4)）。この操作は MPI_Allreduce を用いることで，元の 2 次元分散への再分散も同時に完了させることができる。

本研究ではプログラムは C 言語で実装し，行列は 2.5 次元分散における各 2 次元平面においてブロック分散とした。また行列やプロセスグリッドは正方であると仮定した実装とした。

３．性能評価

Oakforest-PACS を用いて我々のプログラムの性能を評価した。Oakforest-PACS はプロセッサ：Intel XeonPhi 7250（Knights Landing アーキテクチャ，1.4GHz，68 コア，1 ノードあたり 1 プロセッサ），メモリ：MCDRAM 16GB + DDR4 96GB，ネットワーク：Intel Omni-Path アーキテクチャ（100Gbps, Full-bisection Fat-tree）からなる，全 8208 ノードのシステムである。

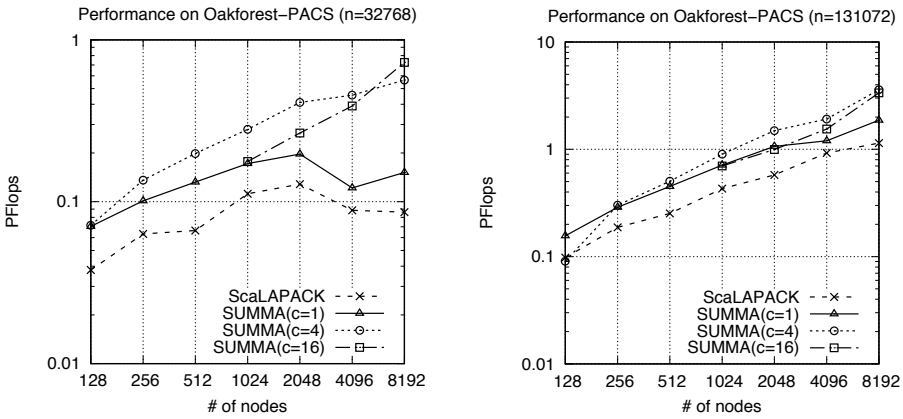
問題サイズを固定しノード数を変化させた場合の強スケーリング性能を評価するため，128, 256, 512, 1024, 2048, 4096, 8192 ノードにおける性能を測定した。2.5 次元アルゴリズムにおける行列のスタックサイズは c=1, 4, 16 とした。なお，c=1 のとき 2D-PDGEMM と等価である。比較のためシステムにインストールされている Intel MKL の ScaLAPACK の性能も測定した（ブロックサイズは 512）。以下に詳細な測定条件を記す：

- KNL のメモリモードは Flat，クラスタモードは Quadrant である
- 使用した Intel ライブラリのバージョンは，Intel compiler 18.0.1，Intel MKL 2018.1，Intel MPI 2018.1.163 であった
- コンパイルは以下のようにして実行した：mpiicc -O3 -xMIC-AVX512 -parallel -qopenmp -lmkl_blacs_intelmpi_lp64 -lpthread -limf -liomp5 -lmkl_scalapack_lp64 -lmkl_intel_lp64 -lmkl_core -lmkl_intel_thread
- ノードあたりの MPI と OpenMP の設定は，256,1024,4096 ノードの時は 4 プロセス/ノード，16 スレッド/プロセスとし，ノード数がそれ以外のときは 2 プロセス/ノード，32 スレッド/プロセスを割り当てた。我々の予備評価の結果では 4 プロセス/ノード割り当てた方が良い性能が得られたが，実装上の総プロセス数の制約により 4 プロセス/ノードにできない場合 2 プロセス/ノードとしている
- ジョブスクリプトの実行オプションは次のように指定した：KMP_AFFINITY=scatter, KMP_HW_SUBSET=1t,I_MPI_FABRICS=tmi:tmi,I_MPI_PIN_PROCESSOR_EXCLUDE_LIST=0,1,68,69,136,137,204,205[2], HFI_NO_CPUAFFINITY=1, I_MPI_PIN_DOMAIN=64, OMP_NUM_THREADS=16, I_MPI_PERHOST=4（ただし前述の通りノード数が 128,512,2048,8192 のとき，OMP_NUM_THREADS=32, I_MPI_PERHOST=2 である）
- 2D 互換 2.5D-PDGEMM における行列の再分散コストは実行時間に含めているが，MPI サブコミュニケータのセットアップコストは含めていない（MPI コミュニケータの構築はライブラリの初期化ルーチンで処理されることを想定しているため）

[2] OS ジッタの影響を避けるためこの設定により 64 コアのみを計算に割り当てている

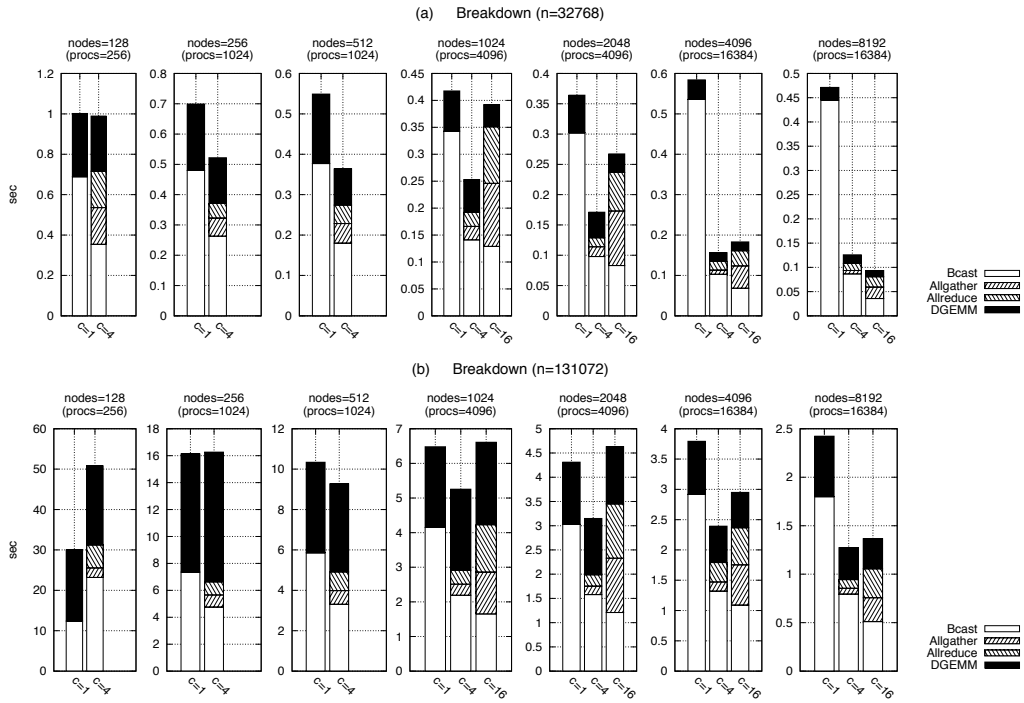
４．結果

第 2 図に行列サイズ n=32768 と n=131072 のときの強スケーリング性能を示す。第 3 図はそれぞれの結果の実行時間の内訳をプロットしたものである。まず第 2 図左の n=32768 のケースでは，2D-PDGEMM（ScaLAPACK の PDGEMM および SUMMA(c=1)）は 2048 ノードを超えたところから性能が横ばいであるが，2.5D-PDGEMM の SUMMA(c=4)および SUMMA(c=16)はそれ以降もスケールしている。しかし SUMMA(c=4)は 2048 ノード以降でスケーラビリティが鈍化し，8192 ノードでは SUMMA(c=16)が SUMMA(c=4)を上回った。これらの性能差が生じる原因は第 3 図の性能内訳より，



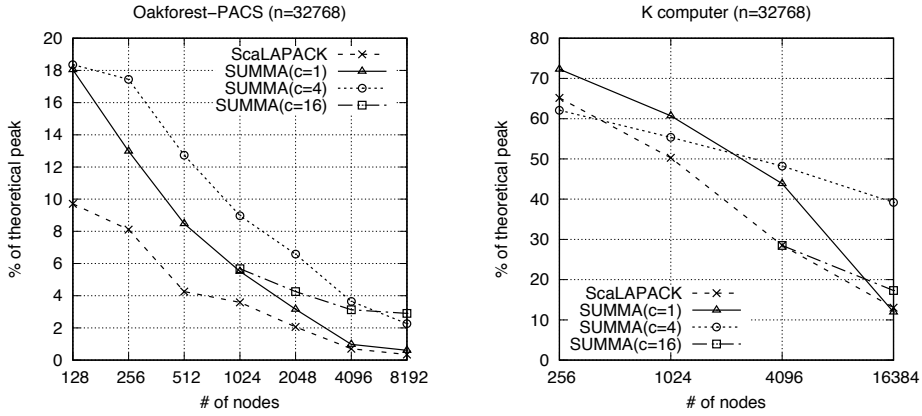
第 2 図:性能（左：n=32768，右：n=131072）

“ScaLAPACK”は Intel MKL の PDGEMM ルーチンの結果，“SUMMA(c=1-4)”は我々の 2D 互換 2.5D-PDGEMM 実装の結果を示している。



第 3 図:実行時間の内訳（上：n=32768，下：n=131072）

2.5 次元アルゴリズムによる Bcast コストの削減と,Allgather による再分散と Allreduce による総和＋再分散コストの増加によって理解できる。一方，第 2 図右の n=131072 の場合，1 ノードあたりの計算サイズが大きくなり，DGEMM の実行時間が全実行時間に対して支配的となり，2.5 次元アルゴリズムの効果は n=32768 の場合と比べると薄くなっている。



第 4 図: Oakforest-PACS（左）と京コンピュータ（右）における n=32768 の場合の理論ピーク演算性能に対する実行効率

第 4 図は，第 2 図左および第 3 図上に示した n=32768 の結果について，縦軸を理論ピーク演算性能に対する実行効率[3]としてプロットし，スーパーコンピュータ「京」で同一の実験を行った時の結果（文献[2]で我々が報告したもの）と並べて示したものである。「京」と比較すると Oakforest-PACS では 2.5 次元アルゴリズムの効果がより大きく現れていることがわかる。両システムにおいては実効性能の違いやプロセス割り当て方式の違い，ネットワークトポロジの違いなどがあるが,理論ピーク性能だけで単純に比較すると,「京」は 20[GB/s]/128[GFlops]≈0.16 である一方，Oakforest-PACS は 25[GB/s]/3046.4[GFlops]≈0.0082 である。したがって Oakforest-PACS は「京」と比べ約 20 倍演算性能に比重を置いた設計であり，その分通信性能がスケーラビリティを阻害する要因となりやすく，2.5 次元アルゴリズムによる通信削減が有効に働いたと考えることができる。

５．まとめ

本評価により，Knights Landing＋Omni-Path ベースのクラスタシステムにおいては，2D 互換 2.5D-PDGEMM が従来の 2D-PDGEMM の性能を上回るとともに，スーパコンピュータ「京」と比べて，2.5 次元アルゴリズムによる通信削減の効果が大きく現れることを確認した。例えば今回の測定では,「京」における評価では確認できなかった 3 万次元程度の行列を 256 ノードで計算するような場合における 2D 互換 2.5D-PDGEMM の有効性が確認できた。「京」と比べるとネットワーク性能が演算性能の割に貧弱と言える Oakforest-PACS のような計算機の特性は近年の主

[3] Oakforest-PACS ではプロセッサの定格最大クロック 1.4GHz で 68 コアが AVX512 で演算を行った場合の性能を理論ピーク演算性能として算出している。ただし我々の測定ではプログラムの実行に 64 コアのみを割り当てている上，AVX512 を使用した場合にクロックが 1.4GHz を下回ることがあるため，理論ピーク演算性能は実現可能な最大性能とは必ずしも一致しない。

流であり，このような計算機において，計算機の規模に対して小〜中規模と言える計算問題の性能を向上させるには，2.5 次元実装が有効であることが再確認された。今後は実アプリケーションに適用しての評価とともに，よりプロダクトレベルに近いルーチンの実装を進める計画である。加えて 2.5D-PDGEMM では行列スタックサイズ（2 次元と 2.5 次元の切り替えも含む）がチューニングパラメータとなるため，性能モデルを構築し，問題サイズと実行環境から最適なスタックサイズを自動選択する手法の構築も今後の課題として挙げられる。

謝　辞

本研究は 2017 年度 Oakforest-PACS スーパーコンピュータシステム「大規模 HPC チャレンジ」による成果である。また本研究は理化学研究所計算科学研究センターフラッグシップ 2020 プロジェクトの一環によるものである。

参　考　文　献

[1] Solomonik, E., Demmel, J.『Communication-optimal parallel 2.5D matrix multiplication and LU factorization algorithms』Proc. 17th International Conference on Parallel processing (Euro-Par 2011), Lecture Notes in Computer Science, Vol.6853, pp. 90-109, 2011

[2] Mukunoki, D., Imamura, T.『Implementation and Performance Analysis of 2.5D-PDGEMM on the K Computer』Proc. 12th International Conference on Parallel Processing and Applied Mathematics (PPAM 2017), Lecture Notes in Computer Science, Vol.10777, pp.348-358, 2018

[3] Van de Geijn, R. A., Watts, J.『SUMMA: Scalable Universal Matrix Multiplication Algorithm』Tech. rep., Department of Computer Science, University of Texas at Austin, 1995