Stratix10 FPGA による大規模な四倍精度行列乗算の性能評価

河野 郁也 [1]
[1] 静岡理工科大学
e-mail : kono.fumiya@sist.ac.jp

1　はじめに

行列乗算は，様々な科学技術計算の根幹に現れる基本的な演算である．計算機による行列乗算は，数値線形代数ライブラリ・BLAS が提供する行列乗算ルーチンである GEMM が著名である．その演算は，$m \times k$ 型および，$k \times n$ 型の実行列 A, B に対して，実スカラー値 α, β を伴って $C = \alpha AB + \beta C$ のように定義される．GEMM ルーチンにおいて，積 AB が主要計算である．$C' = AB$ としたとき，その (i, j) 成分 C'_{ij} は，式 (1) により計算される．式中の p は，$0 \leqq p < k$ の範囲の添字を表す．

$$C'_{ij} = \sum_{p=0}^{k-1} A_{ip} \times B_{pj}, \tag{1}$$

数値計算においては，解の精度が重要視される．行列乗算自体の精度は，それを根幹とする数値アプリケーションの精度にも強く影響する．IEEE754 形式の浮動小数点表現には，単精度や倍精度も他，四倍精度や半精度がある．近年のプロセッサやアクセラレータは，AI・機械学習への需要から半精度などの少ビット数のものをサポートしているが，四倍精度などの大きなものをサポートするものは限られている．一方で，数理最適化問題の 1 つである半正定値計画法 [1] のように，多倍長演算を必要とする問題もある．それらは，一般的な倍精度演算を利用する時に比べて，四倍精度演算を利用することにより数百倍以上の演算時間を要するため，四倍精度演算そのものの高速化も重要である．

我々は，FPGA をアクセラレータとして活用し，四倍精度行列乗算の高速化と，その応用について探求してきた．我々のこれまでの実装 [2] では，正方行列かそれに近い形状の行列同士の積において，高い高速化の効果を得られている．本稿では，筑波大学で現在稼働中である，GPU/FPGA 混載のスーパーコンピュータシステム Cygnus に搭載されている Intel Stratix 10 GX 2800 FPGA を利用して，我々の実装した回路による行列サイズの大きな乗算を行い，その性能を評価する．

2　FPGA アクセラレータ実装の概要

FPGA による数値計算の高速化には，演算のパイプライン化を意識した実装が不可欠である．特に，行列乗算を FPGA で効率的に行うには，式 (1) に含まれる積和演算を行う処理要素 (PE) を並べ，入力モジュールから来るデータを処理しつつ，隣接する PE に送信するという構造を持ったシストリックアレイを構築することが重要である．図 1 は，$P_c \times P_r$ で PE を配置した 2 次元シストリックアレイ構造を表している．行列 A, B をメインメモリから `Read` モジュールを介して読み取り，`Feed` モジュールを通じて PE に送る．データを受け取った PE は，各クロックサイクルごとに隣接する PE に転送する．各 PE が同じ C' の要素に対する積和演算の結果を累積して `Drain` モジュールに送ることで，`Store` モジュールが最終結果をメインメモリに書き戻す．

我々は，FPGA 向け線形代数ライブラリとして開発されている FBLAS [3] の実装を基にして，シストリックアレイを利用した GEMM ルーチンの回路設計を実装した．PE で行われる積和演算は，Nakasato ら [4] が実装した四倍精度演算ユニットを当てはめることで実現している．

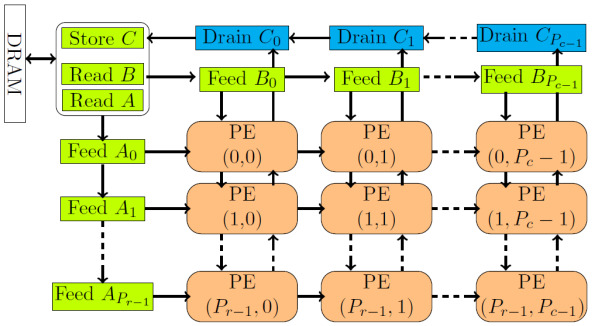

図 1. 行列乗算のための 2 次元シストリックアレイ構造

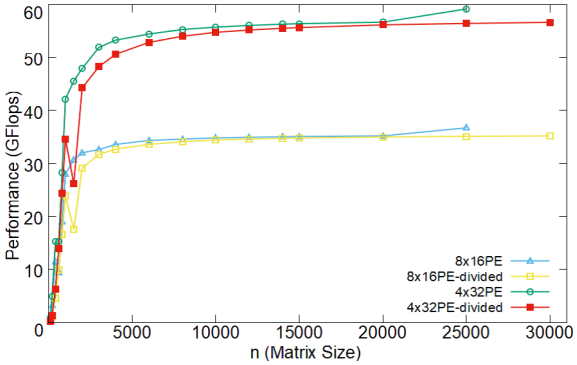

図 2. Stratix10 による行列乗算の演算性能

3 性能評価の概要

Cygnus には FPGA を搭載した 32 のノードがあり，各ノードが Stratix 10 を 1 つ搭載する Nallatech 520N ボードを 2 枚ずつ搭載する．ここでは，そのうちの 1 枚のボードを利用して計算を行った結果を示す．なお利用可能なメモリサイズは，ホスト側が 182GB, FPGA ボード側が 32GB である．図 2 は，Stratix10 FPGA で利用できるハードウェア資源を最大限使用して論理合成した 8×16PE，4×32PE のシストリックアレイ実装を用いて，n 次正方行列 A, B の積の演算性能を，n を 30000 まで増やして調べたものである．凡例中に`-divided` の接尾辞のあるものは，入力行列を 4 分割して，4 回の GEMM ルーチン呼び出しにより計算するものである．

シストリックアレイによる演算は，全ての PE に十分な演算データが行き渡らない状態では性能を発揮できず，$n < 500$ では数 GFlops 程度の性能に留まる．n の増加に応じて向上し，$n = 8000$ 以降でピークに達する．PE の総数は 128 であるが，縦長に配置した 4×32PE の方が性能は高く，8×16PE では 36Flops，4×32PE では 56GFlops となった．行列を分割して演算すると，1 回の GEMM ルーチンが扱う行列サイズが更に小さくなるので，分割しない場合に比べて若干の性能低下があるが，ボードメモリサイズの制限下でより大きな n で計算するために有用であると考えられる．

謝辞

本研究は JSPS 科研費 JP23K11133 の助成を受けたものです．また，本研究成果は筑波大学計算科学研究センターの学際共同利用プログラム（Cygnus）を利用して得られたものです。

参考文献

[1] M. Yamashita, K. Fujisawa, M. Fukuda, K. Kobayashi, K. Nakata, and M. Nakata, Latest Developments in the SDPA Family for Solving Large-Scale SDPs, pp.687–713, Springer US, Boston, MA, 2012.

[2] F. Kono, N. Nakasato, and M. Nakata, “Accelerating 128-bit Floating-Point Matrix Multiplication on FPGAs,” arXiv e-prints, p. arXiv:2306.04087, Jun. 2023.

[3] T. De Matteis, J. de Fine Licht, and T. Hoefler, “Fblas: Streaming linear algebra on fpga,” Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, SC’20, IEEE Press, 2020.

[4] N. Nakasato, H. Daisaka, and T. Ishikawa, “High performance high-precision floating-point operations on fpgas using opencl,” International Conference on Field-Programmable Technology (FPT), pp.262–265, 2018

MN-Core™ と開発環境

坂本 亮[1]，野村 昴太郎[1]
[1]株式会社 Preferred Networks
e-mail: telmin@preferred.jp ， subarutaro@preferred.jp

1　はじめに

近年，大規模言語モデル（LLM）の発展により，AI 技術は社会のさまざまな分野で応用が進んでいる．この技術進展の中で，人工知能の計算処理を効率よく高速に実行する専用ハードウェアである AI アクセラレータは特に注目を集めている．株式会社 Preferred Networks(以下，PFN)では，高性能かつ低消費電力にフォーカスした AI アクセラレータである MN-Core シリーズ[1]を神戸大学と共同で開発している．MN-Core シリーズは，AI アクセラレータでありつつも，AI だけに限らず HPC ワークロードにおける適用も十分可能であるアーキテクチャとなっている．本稿では，MN-Core アーキテクチャとその開発環境について述べる．

2 MN-Core について

2.1 MN-Core アーキテクチャについて

第一世代の MN-Core と第二世代の MN-Core 2 は，それぞれ 2019 年と 2022 年にローンチされた．MN-Core 2 に関する諸元を 表 1 に示す．

MN-Core シリーズで採用している MN-Core アーキテクチャは特徴としてチップ内分散メモリ型アーキテクチャを採用し，シリコン面積中に占める演算に寄与する回路を最大化する設計を取っている．そのため，キャッシュや分岐予測など直接演算に寄与しない回路を省き，ソフトウェアによる最適化を前提としている．プロセッサ上の全ての Processing Element (PE)はプログラムカウンタや命令でコーダを持たずホストで生成された命令列を直接受け取り完全に同期的・並列に SIMD 動作する．これらの特徴により，MN-Core シリーズは高い電力あたりの計算性能を実現している．

MN-Core シリーズは上から DRAM/PDM(ホストインターフェース)，L2B，L1B，MAB からなる階層構造をとっている．L2B および L1B はそれぞれメモリと Reduction Unit からなり，上位/下位階層と分配・放送・縮約などのデータ転送を行う．MAB は 4 つの PE と行列演算ユニット(MAU)からなり，各 PE は ALU とローカルメモリ，汎用レジスタファイルを持ち，メモリから読み出したデータを ALU や MAU に送り演算を行う．DRAM/PDM と L2B，L2B と L1B，L1B と MAB の間のデータ転送命令と PE での ALU と MAU へのそれぞれの演算命令は全て同時発行が可能であり，ソフトウェア側で命令の順序などを考慮してデータ転送と演算を並列に行うようにすることが MN-Core の性能を引き出す上で重要となる．

表 1　MN -Core 2 諸元．性能の単位は全て TFLOPS で左側が行列ベクトル積和算モード，右側がベクトルベクトル積和算モード利用時．

製品名	製造プロセス	FP64 性能	FP32 性能	FP16 性能	消費電力
MN-Core 2	TSMC N7	12.3 / 3.07	49.2 / 12.3	393 / 24.6	300 W

2.2 MN-Core 開発環境について

本節では，PFN が開発している，二つの MN-Core の深層学習向け開発環境および HPC

向け汎用開発環境について述べる.

2.2.1 MLSDK (Machine-Learning SDK)

MN-Core シリーズは AI アクセラレータである. そのため, 第一に深層学習向けの SDK である MLSDK を開発している. MLSDK ユーザーコードの改変を可能な限り減らすこと, MN-Core の性能を活かし切ることを念頭において開発が行われている.

MLSDK は, 深層学習用フレームワークとして広く利用されている PyTorch をインタフェースとして使用している. また, GPU を利用して研究開発を行っているユーザーからの移行を容易にするためのソフトウェアも用意している.

MLSDK はコンパイラとランタイムからなる. コンパイラは PyTorch で定義された ONNX モデルの計算グラフから MN-Core 命令列を生成する. 問題を抽象度に応じて L3IR, L2IR, L1IR の 3 つに分割しており, コンポーネント単位でのアルゴリズムの改善が可能になっている. ランタイムは, MN-Core 上の DRAM とホストのメモリとの計算に必要なデータのやり取りを行い, コンパイラで生成された命令列を MN-Core に送ることで計算を実行する.

2.2.2 HPCSDK (High-Performance Computing SDK)

MLSDK を用いても PyTorch のインターフェースを利用してある程度汎用的なコードを書くことは可能であるが, MN-Core シリーズの演算性能を HPC を含む汎用計算分野においても利用するために, より従来の HPC アプリケーション開発環境に近い HPCSDK を並行して開発している. HPCSDK は, それぞれ OpenCL[1]および OpenACC[2]ライクな言語処理系である MNCL と MNACC とそのコンパイラ, デバイスドライバとユーザコードの中間に位置する HPC SDK ベースランタイム, 各種ライブラリ, プロファイラやデバッガなどのアプリ開発ツールからなる.

MNCL および MNACC を用いて書いたコードのカーネル部分はそれぞれのコンパイラによって MN-Core 向けの命令列に変換され, MLSDK と同様ベースランタイムを通じて命令列やデータのやり取りを行う. HPCSDK でも L1IR 部分は MLSDK と共通して使っており, PE レベルでの最適化は同様に行われる.

3 まとめ

本稿では, MN-Core アーキテクチャと, MN-Core シリーズ向けの開発環境について述べた. PFN では, 深層学習向け開発環境として MLSDK を, HPC 向け開発環境として HPCSDK をそれぞれ継続して開発し, 深層学習分野でも, HPC 分野でも, MN-Core の高い演算能力を利用可能にしていく考えである.

謝辞 この成果は, ＮＥＤＯ（国立研究開発法人新エネルギー・産業技術総合開発機構）の委託業務（JPNP16007）の結果得られたものです. また, 本研究は, 文部科学省「次世代計算基盤に係る調査研究」事業の助成を受けたものです.

参考文献

[1] https://projects.preferred.jp/mn-core/
[2] https://www.khronos.org/opencl/
[3] https://www.openacc.org

