B-020

Jetson AGX Xarvier における拡張ベクトル化 LU 分解法の評価
Evaluation of Extended Vectorized LU Decomposition Method in Jetson AGX Xarvier

土屋 広記[1)] **富永 浩文**[2)] **中村 あすか**[3)] **前川 仁孝**[3)]
Hiroki Tsuchiya Hirobumi Tominaga Asuka Nakamura Yoshitaka Maekawa

1 はじめに

拡張ベクトル化 LU 分解法 (EMVA) は，命令レベル並列性を抽出することで，ランダムスパース方程式を高速に求解するアルゴリズムである [1]．EMVA は，LU 分解法の並列性から，CUDA を用いて実装されている [2]．CUDA を用いた EMVA(cuEMVA) には，条件分岐を回避するために CPU と GPU の演算資源を併用する手法が提案されている．本手法は，CPU と GPU で演算結果を共有するため，CPU-GPU 間のデータ転送部分のチューニングが重要となる．CPU-GPU 間のデータ共有方式に，Global Memory(GM) に加えて，Unified Memory(UM) や Zero Copy Memory(ZCM) の 3 種類がある．Jetson AGX Xavier(Xavier) では，GM と比較し，UM や ZCM の機能で CPU-GPU 間のデータ転送を高速化できる．ただし，どのメモリ機能が Xavier での EMVA の実行速度に有用かは明らかでない．そこで，本稿では，cuEMVA を高速に実行する通信方式を明らかにするため，Xavier を用いた cuEMVA で 3 種類のデータ転送方式を使用し，各方式の有用性を評価する．

2 拡張ベクトル化 LU 分解法 (EMVA)

EMVA は，実行レベルを利用し，高い並列性のある命令を抽出することで，LU 分解の求解を高速に行う手法である [1]．図 1 に EMVA の命令で並列性を抽出する動作の例を示す．図 1 中の演算命令は，LU 分解の積差演算や，除算演算である．本手法は，LU 分解の求解で必要な演算をメモリ上に書き出し，各演算に必要な実行レベルを設定する．同一実行レベルの演算は，同時に実行可能であり，高い並列性を持つことが知られている．この高い並列性に着目し，高い並列性をもつ CUDA を用して高速化を実現した手法に cuEMVA が提案されている [2]．CUDA を利用した cuEMVA 法は，CUDA の特性に合わせた最適化手法が重要である [2]。以下，2.1 節では cuEMVA 手法，2.2 節では CUDA 環境における CPU-GPU 間のデータ共有方法について述べる。

2.1 cuEMVA

cuEMVA は，実行レベルごとに除算を計算する CUDA カーネルと積差を計算する CUDA カーネルを起動する．本手法は，同じレベルに設定した演算を CUDA カーネルで同時に実行する．CUDA は，1 ワープ (32 スレッド) ごとに命令を実行する．CUDA の条件分岐は，両方の分岐先の命令を実行するワープダイバージェンスが発生し，実行時間が遅くなるとが報告されている [3]. このため，cuEMVA の CUDA カーネルでは，指定した演算数

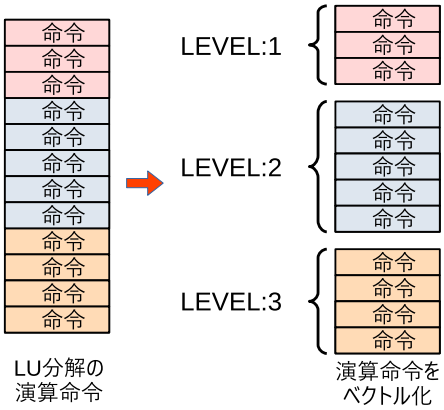

図 1 拡張ベクトル化 LU 分解法の実行レベルの抽出

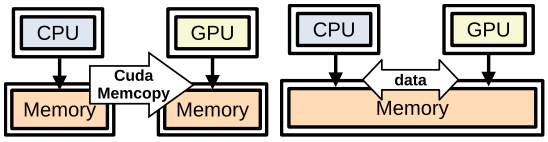

図 2 Global Memory

図 3 Unified Memory

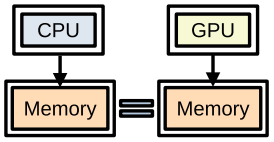

図 4 Zero Copy Memory

に満たない場合，スレッド番号を利用した条件分岐により，実行可能な演算数よりも多くの演算を行わないように制御する．ワープダイバージェンスは，32 の倍数で実行可能な演算を CUDA カーネルで実行し，残りの演算を CPU で実行することで解決する CPU を併用する実装がある．この手法は，CPU でも演算を割り当てることができるが，演算結果を共有する必要があるため，CPU-GPU 間の通信を減らすデータ転送部分のチューニングが必要である．

2.2 CPU-GPU のデータ共有

CUDA で CPU-GPU 間のデータを共有する方式には，明示的にデータ転送を指定する GM やシステム側で自動的に共有する UM，ZCM がある [4]．GM は，CPU-GPU 間の転送命令である cudaMemcpy で，CPU からのデータを自分のメモリにコピーする．図 2 に，GM のデータの共有方法を示す．図 2 中の矢印はデータのアクセスを表す．

GM は，プログラムで CPU-GPU 間のデータのアドレスが対応する場所とデータの転送方向を cudaMemcpy で指定することでデータを保存する．GM を使用する

1） 千葉工業大学大学院情報科学研究科情報科学専攻 Graduate School of Information and Computer Science, Chiba Institute of Technology
2） 千葉大学情報戦略機構 Chiba University Digital Transformation Enhancement Council
3） 千葉工業大学情報工学科 Department of Computer Science, Chiba Institute of Technology

表 1　Jetson AGX Xavier の実行結果 [ms]

問題名	行列サイズ	要素数 [10^3]	演算数 [10^3]	命令数 [10^3]	Unified Memory		Zero Copy Memory		Global Memory	
					併用無し	併用あり	併用無し	併用あり	併用無し	併用あり
memplus	17,758	126	870	1	382.8	196.7	127.7	43.8	124.1	460.9
mult_dcop_01	25,187	193	2,291	22	8089.3	372.7	2727.8	93.1	2730.1	1172.0
circuit4	68,902	375	6, 809	2	5454.2	608.8	712.1	115.3	1676.4	1786.2
bcircuit	80,209	307	3, 221	14	1152.2	894.9	309.6	108.5	170.1	4171.8
hcircuit	110,355	787	2, 782	4	841.9	459.0	302.7	107.1	310.4	2211.8
rajat23	105,676	630	1, 970	2	1697.0	594.4	548.7	119.5	547.9	2439.6

cuEMVA は，カーネル起動をする前に演算に必要なデータを転送する．

UM や ZCM は，CPU と GPU で対応するアドレスを指定することで，同一のメモリ空間としてアクセスできる．図 3，図 4 に，UM と ZCM のデータの共有方法を示す．UM は，CPU と GPU のメモリを共有する環境として扱うことができる．UM の実際のメモリ空間のデータは，CPU や GPU の演算に応じてシステム側が自動でデータを転送する．一方，ZCM の実際のメモリ空間のデータは，CPU 側のメモリに保存する．

3　評価

本稿は，Xavier を用いた cuEMVA で，高速化に有用なデータ転送方式を評価する．Xavier は，CPU と GPU で共有するメモリを持つ．このため，UM や ZCM を使用することで，GM よりも高速に求解できると考える．また，CPU を併用する手法は，同期処理により，CPU-GPU 間でデータ転送の処理回数が減少する．このため，UM や ZCM の機能は，高速化に有用である．

Xavier を用いた cuEMVA で，有用なデータ転送方式を明らかにするため，各データ共有方式の使用時の実行時間を測定する．評価方法は cuEMVA で CPU を併用する場合としない場合で，GM，UM，ZCM のデータ共有方式の実行時間を比較する．cuEMVA と CPU を併用する手法のスレッドブロックサイズは，Nvidia Occupancy Calculater でワープの占有率が高い 256 に設定する．対象問題は，回路情報から生成された Florida Sparse Matrix Collection の回路方程式とし，実行時間は 10 回の平均で算出する．評価環境の Jetson AGX Xarvier は，メモリ 32GB，GPU に 512 個の cuda コアを搭載し，CUDA version は 11.4 である．

表 1 に対象問題と，Jetson AGX Xarvier を用いた CUDA カーネルの実行時間を示す．表 1 の要素数は非ゼロ要素数，演算数は EMVA の演算数，命令数は，ベクトル化する命令の数を示す，実行結果で最も高速なデータ共有方式は，CPU を併用する場合の ZCM である．UM と ZCM は，cuEMVA から CPU を併用することにより，全ての問題で実行速度が向上する．これは，CPU と GPU のデータの共有を同一のメモリアドレスで行い，CPU-GPU 間のデータ共有に時間がかからないためであると考える．

そこで，ZCM の高速化理由を明らかにするため，NVIDIA Nsight Systems[5] を使用し，CPU と GPU がカーネル起動を行う場合に実行する命令を調査する．解析に使用する対象問題は，各手法での実行時間の差が大きい CPU を併用する場合の bcircuit を対象とする．図 5 に，NVIDIA Nsight Systems による各データ共有方式のプログラムの解析結果を示す．図中の命令は，GPU が発行する命令の実行時間である．また，縦線は，1 カーネル

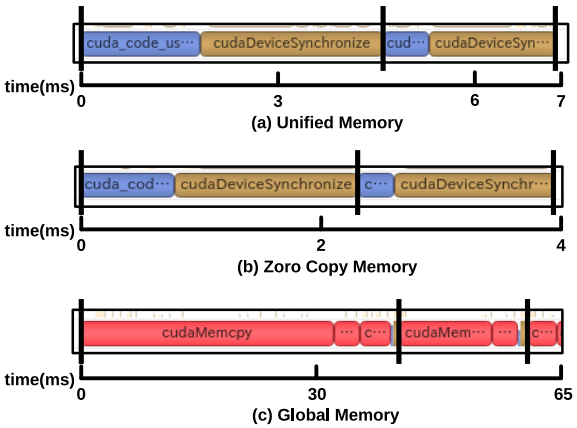

図 5　Nvidia nsight systems の解析結果

あたりの実行時間の間隔である．図 5(a) の UM と図 5(b) の ZCM を比較すると，UM は，ZCM より 1 カーネルあたりの実行時間や同期時間が長い．また，図 5(c) の GM は，UM や ZCM より 1 カーネルあたりの実行時間が長い．これは，cudaMemcpy の実行時間が長いためである．このため，Xavier は cuEMVA を求解する場合に，ZCM が最も有用なデータ共有方式であると考えられる．

4　おわりに

本稿は，Xarvier で EMVA の求解に有用なメモリ管理機能と手法を評価するために回路方程式のベンチマーク問題である Florida Sparse Matrix Collection の回路方程式をデータ共有方式ごとに求解した．評価の結果，Xarvier を用いた EMVA は，ZCM を使用することで，CPU の併用による演算資源の共有が高速化に有用であることを示した．

参考文献

[1] 鹿毛哲郎，下郡慎太郎：ベクトル計算機による高速回路解析のためのベクトル化処理技法，電子情報通信学会論文誌 D, Vol.70, No.8, pp. p1597–1606 (1987).

[2] 富永浩文，中村あすか，北川翔一，前川仁孝：拡張ベクトル化 LU 分解法による回路方程式求解のための CUDA カーネルの実装方式の評価，電気学会研究会資料. ECT= The papers of technical meeting on electronic circuits, IEE Japan/ 電子回路研究会 [編]，Vol. 2022, No. 45-58, pp. 69–71 (2022).

[3] 富永浩文，前川仁孝：CUDA によるランダムスパース方程式求解の命令レベル並列性を用いた高速化手法，情報処理学会論文誌プログラミング (PRO)，Vol. 7, No. 1, pp. 10–17 (2014).

[4] Cheng, J., Grossman, M., McKercher, T. et al.: CUDA C プロフェッショナルプログラミング，(2015).

[5] NVIDIA Nsight Systems,”https://developer.nvidia.com/nsight-systems” (accessed 2023-06-19)