5L-06

GPU 上における線形ソルバの高速化と評価

石川 渉† 藤井昭宏† 田中輝雄†

工学院大学†

1 はじめに

連立一次方程式の求解は，多くの数値シミュレーションに現れ，計算時間の大部分を閉める[1].そのため大規模で複雑な数値シミュレーションを行うためには，連立一次方程式の高速なソルバが必須である．近年では GPU による連立一次方程式ソルバの高速化が図られている．

本研究では，NVIDIA の Pascal/Volta 世代の GPU 上でのベクトル内積および疎行列ベクトル積と共役勾配法（CG 法）を対象とする．この 3 種類に対し，Block/Thread 分割数の考慮，ELL 形式によるコアレスアクセス，GPU 世代における変更点であるスレッド同期の違いに基づいた各 nvcc コンパイラオプションによる実行時間および実行性能の調査を行った．

2 Volta 上でのオプションについて

実装にあたり，以下の環境を使用する．

表 1　東京大学基盤センター「Reedbush-h」の性能表

CPU		GPU	
Intel Xeon E5-2695v4 2.1GHz		NVIDIA Tesla P100	
コア数	18	Cuda cores	3,584
メモリ	256 GB	SM	64

表 2　産総研「abci」の性能表

CPU		GPU	
Intel Xeon Gold 6148 2.4GHz		NVIDIA Tesla V100	
コア数	18	Cuda cores	5,120
メモリ	256 GB	SM	80

＊256 thread/block

NVIDIA Tesla V100 [2] は Pascal 世代までの GPU で行われてきた暗黙のワープ単位での同期が存在しないという違いが存在する．そこで，本来のコンパイルオプションである-gencode arch = compute_70 , code = sm_70 ではなく，-gencode arch=compute_60,code=sm_70 に変更し実行することによって従来までの GPU と同じような同期を Volta でも行う．今回は，Volta の GPU で-gencode arch = compute_60 のオプションでの実行と -gencode arch = compute_70 のオプションでの実行の 2 通りを評価対象とする．本実験では，SuiteSparce Matrix Collection [3]の行列を使用した．

Acceleration and Evaluation of Linear Solver on GPU
Wataru Ishikawa†, Akihiro Fujii†, Teruo Tanaka†
†Kogakuin University

3 数値実験

3. 1 ベクトル内積

内積演算を行うにあたり，グローバルメモリとシェアードメモリを使用する．シェアードメモリはグローバルメモリよりも高速なメモリだが，Block 内の thread のみ共有可能であり，Block 間での共有が不可能である．積和演算は，ブロックサイズごとに部分和を求め，集約する．本稿では，この演算を OpenMP によるスレッド並列を行った CPU・GPU（Volta/Pascal）の 3 通りで実装し，計測を行った．

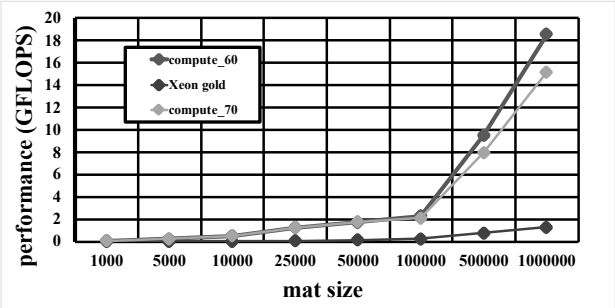

図 1　ベクトル内積性能評価

図 1 では，内積演算では GPU において CPU の約 10~11 倍の性能向上が確認された．GPU のコンパイルオプション compute_70 (Volta) は，compute_60 (Pascal) に比べ約 25%の性能向上を確認した．

3.2 行列ベクトル積

本実験では，行列ベクトル積の対称を CG 法の前提条件である対称正定値行列とした． GPU 上のグローバルメモリは連続アクセス時の転送速度は高速である一方，レイテンシが大きくランダムアクセス性能が低い．そこで今回，ELL 形式 (図 2) を採用する．連続方向が縦方向になるように格納し，thread の割り当てを連続させコアレスアクセスを実現し，行単位で thread 並列を行った．

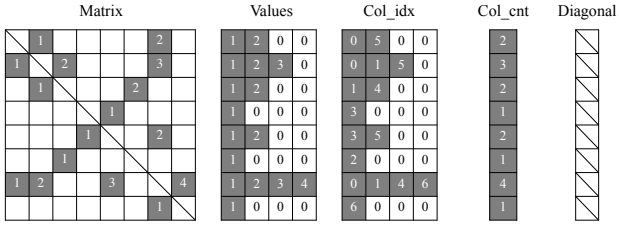

図 2　ELL 圧縮方式

Pascal GPU と Volta GPU (compute_60, compute_70) , OpenMP によるスレッド並列化を行った CPU (Xeon gold) の 4 通りで性能評価を行った．以降のグラフは，行列が横軸になっているが，非ゼロ要素の小さい順にソートを行っている．

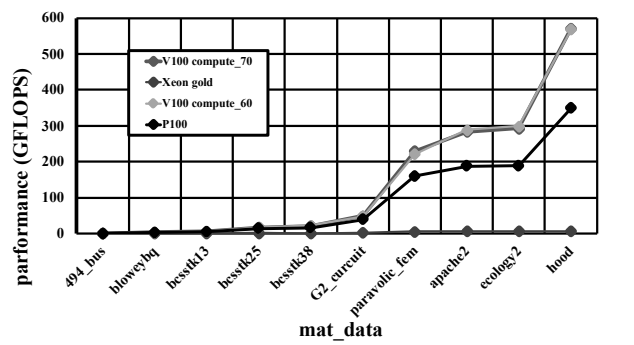

図３　行列ベクトル積性能比較

図３より，行列ベクトル積では Volta の GPU で Pascal の最大 63%の性能向上が確認された．また，compute_60 および compute_70 については違いが見られなかった．

3.3 Thread/Block 考慮

GPU を用いた高速化では，Streaming Multiprocessor (以下，SM) の数やブロックサイズ，occupancy (1SM あたりの shared memory と Resister 占有度) を考慮する必要がある．GPU には 32thread 単位で同時実行される Warp という単位が存在する．ブロックサイズを指定する際，Warp 単位でないと割り切れない分，他の演算器が休んでしまい，性能低下が発生する．そのためブロックサイズは 32 の倍数が望ましい．また，occupancy を高めるために，各 SM を多数のブロックで占有するため，グリッドサイズを指定する際は SM の数よりも十分に大きくする必要がある[4].

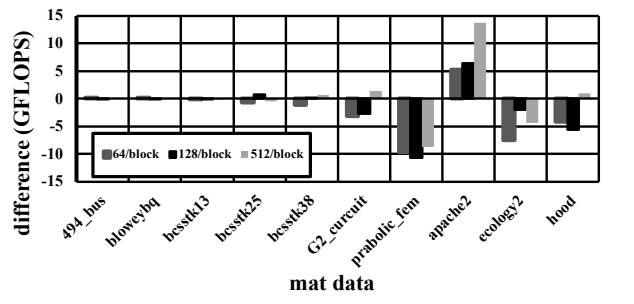

図４　ブロックサイズ 256 との性能差分

ブロックサイズを変更して計測し，図４を得た．図４より，行列 G2_cursuit より非ゼロ要素数が少ない行列では影響が少ないものの，G2_curcuit 以上の行列については実行時間の変化が見られた．行列 apache2 では約 10GFLOPS 性能が向上した．一方で，行列ベクトル積では性能変化の割合としては約 5%であった．

3.4 CG 法

共役勾配法 (CG 法) は対称正定値行列を係数とする連立一次方程式を解くための反復解法である．本実験で扱う問題は，条件数が大きい行列が多く，収束が難しいため，反復回数を 100 回に固定して計測を行った．また，CG 法では GFLOPS の算出が難しいため，性能評価には実行時間を用いた．CG 法を，Pascal GPU と Volta GPU (compute_60 ，compute_70)の 3 通りで計測を行った．

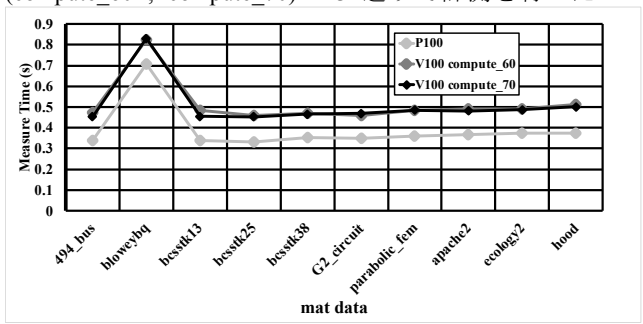

図５　CG 法性能比較

CG 法を実装し，図５を得た．Volta GPU の性能が Pascal GPU を下回る結果となり，また Volta 上では compute_60 と compute_70 の違いはほとんど見られなかった．

4 おわりに

本研究では，Pascal GPU および Volta GPU の性能比較を内積演算，行列ベクトル積，共役勾配法について行った．内積演算と行列ベクトル積は，Pascal GPU に対し Volta GPU はそれぞれ最大 25%，最大 63%の性能向上を確認した．また GPU のスレッドブロックサイズを変更して計測を行い，行列ベクトル積では約 5%の変化が確認できた．また共役勾配法では，Volta GPU の性能が Pascal GPU を下回った．ベクトル内積と行列ベクトル積との性能比較と反する結果となっており，今後，NVIDIA Visual Profiler を用いて演算ごとに性能をプロファイルし，原因を追究していく必要がある．Volta 上でのオプションについてはベクトル内積では compute_60 が compute_70 よりも性能を発揮していたが，行列ベクトル積と CG 法に関しては大きな違いが見られなかったため，compute_60 での実行が有効であると考えられる．

謝辞

本研究の一部は JSPS 科研費 JP18K19782 , JP18K11340 , JP15K15998 の助成を受けたものです．

参考文献

[1] 高橋光佑，藤井昭宏，田中輝雄，“マルチ GPU を用いた AMG 法”，情報処理学会，6 号，no.29，pp.1-7, 2012 (2018/8/25 参照)

[2] NVIDIA TESLA V100 GPU ARCHTECTURE(2018/10/3 参照)
https://images.nvidia.com/content/volta-architecture/pdf/volta-architecture-whitepaper.pdf

[3] SuiteSparceMatrixCollection(2018/8/27 参照)
https://sparse.tamu.edu/

[4] プログラムを高速化する話Ⅱ(2018/12/5 参照)
https://www.slideshare.net/KMC_JP/gpgpu-91122680