2H-01

CUDAにおけるダイナミックパラレリズムを用いたJDS形式疎行列ベクトル積の評価

長坂　一生†　　富永　浩文‡　　中村　あすか††　　前川　仁孝†

† 千葉工業大学情報科学部情報工学科　　‡ 千葉工業大学大学院　情報科学研究科情報科学専攻

†† 千葉工業大学専門研究員

1　はじめに

疎行列ベクトル積（SpMV）は，多くの科学技術計算で用いられるため，CUDAによる高速化が行われている．CUDAを用いたSpMVに有効な疎行列圧縮型式のひとつに，JDS(Jagged Diagonal Storage) 形式がある [1]．CUDAにおけるJDS型式を用いたSpMVは，1ブロックをワープの倍数に設定し，1スレッドに1行を割り当て計算する．1ワープに割り当てられた行の非ゼロ要素数が異なる場合，分岐処理によるワープダーバージェンスのオーバーヘッドが発生する．本オーバーヘッドは，同じ要素数の行のみを計算するカーネルを生成することで削減できる．そこで，本稿では，JDS形式を用いたSpMVに対してダイナミックパラレリズム (DP:Dynamic Parallelism) を用いて新たにカーネルを生成する手法を提案し，その効果を評価する．

2　JDS形式を用いたSpMV

JDSは，CUDAを用いた疎行列演算において行列を圧縮することで，冗長計算とメモリアクセス回数を削減する．図1にJDS形式で格納した疎行列の例を示す．図1(a)の疎行列をJDS形式に圧縮した形が図1(b)である．図1(b)中のvalは行列の非ゼロ要素の値，colは非ゼロ要素の列番号，ofsは列のオフセット，rowは非ゼロ要素数でソートした行番号，maxは最も非ゼロ要素数が多い行の非ゼロ要素数である．valの要素の格納順番は図1(c)のように列方向優先である [1]．

JDS形式SpMVは，ワープ内のスレッドが連続したメモリへアクセスするとき，コアレスアクセスとなりアクセス回数を削減できる．このため，JDS形式SpMVは，ワープ内のスレッドのメモリアクセスが連続になるように1スレッドに1行の計算を割り当てる．

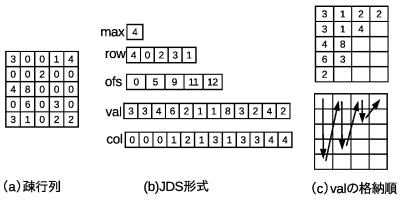

図 1: JDS 形式への圧縮

3　DPを用いたJDS

JDS形式を用いたSpMVは，ワープ内の各スレッドに割り当てられた行の非ゼロ要素数が異なる場合，ワープダーバージェンスによるオーバーヘッドが発生する．このオーバーヘッドは，非ゼロ要素数の差が大きさに比例する．このため，CUDAにおけるJDS形式SpMVは，1ワープに非ゼロ要素数の同じ行を割り当てる．そこで提案手法では，DPを用いて同じ要素数の行のみを計算するカーネルを生成する．

3.1　DP

DPは，カーネル (親カーネル) 上から新たなカーネル (子カーネル) を起動する機能であり，Kepler以降のNVIDIA製のGPUに搭載されている．図2にDPの例を示す．図2は，親カーネルの各ブロックのスレッド0が子カーネルを生成する例である．子カーネルのグリッドおよびブロックサイズは，カーネルごとに設定できるため，本例では異なるグリッドサイズのカーネルを起動している．カーネル上からカーネルを起動させることにより，ホスト– デバイス間の制御を削減するだけでなく，ワープダーバージェンス発生の抑制や，SM占有率の向上が期待できる [2]．

3.2　縦分割によるカーネル生成

縦分割手法は，要素数の同じ列ごとに行列を分割し，分割した列の計算を子カーネルに割り当てる．図3(a)に縦分割の例を示す．図3(a)は，行列を縦方向に4つに分割し，カーネルに割り当てる例である．本手法は，階層的に子カーネルを起動するため，図3(a)において

Evaluation of JDS-Based Sparse Matrix Vector Multiplication Using Dynamic Parallelism on CUDA

†Nagasaka Issei　‡Tominaga Hirobumi　††Nakamura Asuka　†Maekawa Yoshitaka

†Information and Computer Science,Chiba Institute of Technology, Narashino, Chiba 275-0016, Japan

‡School of Information Technology and Electronics, Tokai University, Narashino, Chiba 275-0016, Japan

††Resercher, Chiba Institute of Technology, Narashino, Chiba, 275-0016, Japan

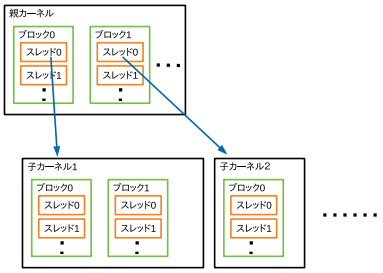

図 2: DP によるカーネル生成

親カーネルのスレッド 1 が子カーネル 1 を，子カーネル 1 のスレッド 1 が子カーネル 2 を起動し計算を割り当てる．すべてのカーネルは，子カーネルを起動した後，自身の親カーネルに割り当てられた列を計算する．本手法では，複数のカーネルが分割して一行を計算するため，積和演算に排他処理を用いる．

3.3 横分割によるカーネル生成

横分割手法は，要素数の同じ行ごとに行列を分割し，分割した行の計算を子カーネルに割り当てる．図 3(b) に横分割の例を示す．図 3(b) は，行列を横方向に 4 つに分割し，カーネルに割り当てる例である．本手法は，まず親カーネルのスレッドに対して要素数が同じ行の行数を割り当てる．親スレッドは，計算をワープごとに割り当てるため，1 カーネルあたり 32 行を計算するように複数の子カーネルを起動し計算を割り当てる．要素数が同じ行を 32 行区切りで子カーネルに割り当てるため，図 3(b) の場合は 2 個の子カーネルに割り当てる．

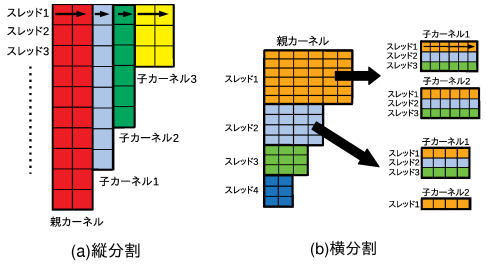

図 3: 提案手法におけるカーネル生成

4 評価

JDS に対する DP の有効性を評価するために，DP を用いない手法と用いる手法で，疎行列ベンチマーク問題である Florida Sparse Matrix Collection [3] の中から 6 問に対して SpMV を行う．評価に使用した問題は，行ごとの非ゼロ要素数に偏りがない 3 問 (1138_bus, gemat11，OPF_10000) と行ごとの非ゼロ要素数に偏りがある 3 問 (pct20stif，ckt11752，OPF_6000) である．評価環境は，CPU が Intel(R) Xeon(R) CPU E5-2687W，GPU が GeForce GTX TITAN Black，CUDA バージョンが V9.0.176 である．表 1 に SpMV の実行時間を示す．表 1 より，横分割手法は，従来手法に比べ，すべての問題で実行時間が増加することが分かる．これは，要素数ごとの行数が少なかったため，各子カーネルに割り当てられる行数が少なく SM の演算リソースを最大限利用できなかったためである．一方，縦分割手法は，ckt11752 と OPF_6000 の 2 問に対して実行時間を削減できることが分かる．縦分割で最も高い高速化率が得られた ckt11752 は，他の問題に比べて非ゼロ要素の偏りが最も大きい行列である．このため，非ゼロ要素が一部の行に偏っている行列に対して，子カーネルでの並列性が維持しやすく，横分割や従来手法に比べダイバージェンスの抑制と SM の占有率を高く維持できると考えられる，また，pct20stif は，非ゼロ要素の偏りが段状にあったため，子カーネルが多く生成され実行時間が増加した考えられる．

表 1: 実行時間 [ms]

問題名	従来手法	縦分割手法	横分割手法
pct20stif	0.539	0.717	18.707
1138_bus	0.108	0.214	39.112
ckt11752	4.205	2.092	17.609
gemat11	0.128	0.257	1.546
OPF_10000	0.216	0.218	9.826
OPF_6000	0.194	0.166	6.814

5 おわりに

本稿では，JDS 形式を用いた SpMV に対して DP を用いる手法を提案し，その有効性を評価した．評価の結果，縦方向に分割する手法において，最大 2.01 倍の高速化率を得られた．縦分割手法は，一部の行に非ゼロ要素の偏りがある行列に対して有効であると考えられる．

参考文献

[1] Cevahir，A.，Nukada，A. and Matsuoka，S. : High performance conjugate gradient solver on multi-GPU clusters using hypergraph partitioning，Computer Science Research and Development，Vol.25，No.1-2，pp. 83–91，(2010).
[2] John Cheng，Max Grossman，Ty McKercher : CUDA C プロフェッショナルプログラミング，pp. 140–152 (2015).
[3] Davis，T. A. and Hu，Y.: The university of Florida sparse matrix collection，ACM Trans. Math. Softw.，Vol. 38，No. 1，pp. 1–25 (2011).