ISSN 2186-5647

P-19

MPS 法における疎行列ベクトル積の GPU による高速化

日大生産工(学部)　○須藤　圭吾
都立産技高専　三浦　慎一郎
日大生産工　　角田　和彦

1 まえがき

数値流体解析手法において，粒子法の一種である MPS 法では流体の非圧縮性を満たすための圧力のポアソン方程式が構成され連立方程式を解く必要がある．この連立方程式は一般的に反復解法が用いられ，このときの疎行列ベクトル積（Sparse Matrix-Vector Multiplication; SpMV）の計算コストが最も大きい．また疎行列ベクトル積では，演算量よりもメモリアクセス量の方が大きいメモリ律速な問題となるためメモリバンド幅の大きい計算機システムが有効となる．このようなことから CPU よりもメモリバンド幅の大きい GPU を利用した計算が注目されている[(1),(2)].

本研究では流体解析のための 3 次元 MPS 法による計算を行うため，GPU を用いて計算コストの大きい疎行列ベクトル積の高速化を行うことを目的とし，大規模計算への見通しを立てることを目指す.

2 MPS法によるポアソン方程式の構成

非圧縮性粘性流体と仮定した流体の質量保存則の連続の式及び運動方程式のナビエストークス方程式を以下のように与える.
圧力のポアソン方程式を組み立てるため, 係数行列には次のラプラシアンモデルを用いる.

$$\langle \nabla^2 \Phi \rangle_i = \frac{2d}{\lambda n^0} \sum_{j \neq i} [(\Phi_j - \Phi_i)\omega(|r_j - r_i|)] \quad (1)$$

となる. ここで分散の増加を解析解と一致させるため，次元数d，係数λである.

3 疎行列の格納方法

3-1. ELLpack形式

ELLpack形式のアルゴリズムについて述べる.

$$A = \begin{bmatrix} 4 & 0 & 0 & 1 \\ 0 & 2 & 0 & 0 \\ 2 & 0 & 6 & 3 \\ 0 & 1 & 0 & 5 \end{bmatrix} \quad (2)$$

ELLpack形式は行列データを2つの配列valとcolindに格納する．rowMaxは行列Aの各行の非零要素数の最大値を表す．倍精度配列valの大きさはN×rowMaxの大きさとなる．行列Aの各行の非零要素を列方向に沿って格納する. 最初の列は各行の最初の非零要素数からなる．格納する非零要素数がない場合，0を格納するため，配列valは

$$val = \begin{bmatrix} 4 & 1 & 0 \\ 2 & 0 & 0 \\ 2 & 6 & 3 \\ 1 & 5 & 0 \end{bmatrix} \quad \text{colind} = \begin{bmatrix} 1 & 4 & 1 \\ 2 & 1 & 1 \\ 1 & 3 & 4 \\ 2 & 4 & 1 \end{bmatrix} \quad (3)$$

となる．また，整数配列colindの大きさはN×rowMaxの大きさとなる．配列valに格納された非零要素の列番号を格納する.

4-2. ELLpack-R形式

ELLpack-R 形式は ELLpack 形式を行と列を入れ替えた形となる．これは CRS(Comressed row storage)形式と同じとなるが, CRS ではゼロ成分を完全に除いて記憶されるのに対し，ELLpack-R 形式では計算量は CRS と同等であるが，記憶容量は ELLpack と同等となる. 最内ループでは行方向へのアクセスとなり, CRS 形式と同様にキャッシュの効果が期待できるため，スカラ並列計算機に向いている.
図 1 に ELLpack の OpenMP コードを示す. ELLpack では内側ループを並列化する.

GPU acceleration of sparse matrix vector product in MPS method

Keigo SUDO, Shinichiro MIURA and Kazuhiko KAKUDA

図２に ELLpack 形式での CUDA 化したコードを示す．１ スレッド当たり出力ベクトルの１要素に相当するものである．

```
#pragma omp parallel
{
    #pragma omp for
    for (int i = 0; i < NA; i++) ax[i] = 0.0;
    for (int j = 0; j < rowMax; j++){
        #pragma omp for
        #pragma simd
        for (int i = 0; i < NA; i++){
            int ii = j*NA + i;
            int col = col_ind[ii];
            ax[i] += a[ii] * x[col];
        }
    }
}
```

図１　SpMV with ELLpack（OpenMP）

```
unsigned int i = blockIdx.x * blockDim.x +
threadIdx.x;
if ( i < NA){
    double tmp = 0.0;
    int rowNum = row[i];
    for (int j = 0; j < rowNum; j++) {
        int ii = j*NA + i;
        int col = col_ind[ii];
        tmp += a[ii] * x[col];
    }
    ax[i] = tmp;
}
```

図２　SpMV with ELLpack（CUDA）

４　ダムブレイク問題による検証

本手法を3次元MPS法によるダムブレイク問題に適用する．GPU計算の有効性を検討するため，CPUによる計算も行う．

本研究の使用する計算環境は，GPU1はNvidia Tesla K40，GPU2はNvidia GeForce TITAN BLACK，CPU計算は Intel Xeon E5-1650 V3 （6コア，12スレッド，L3-cache 15MB）を用いる．CUDAバージョンは7.5を用いる．演算時間CPUでは演算回数を粒子数（未知数）×最大行数とし，GPUではゼロ成分を除いた演算回数で，一回の疎行列ベクトル積の時刻を平均から算出している．図3はダムブレイク問題の総粒子数269,205，圧力計算粒子数が160,578において，時刻が0.4[sec]での状態を示している．表１にCPUとGPUでの計算結果を示す．表１より，粒子数の比較的小さい計算ではCPUとの差が大きくないが，粒子数が増えるに連れ，両者の差が顕著となり，GPUの有効性がみられている．

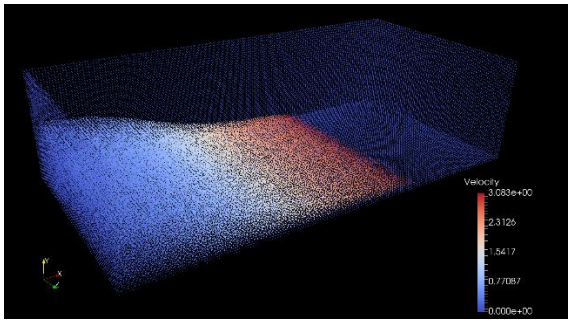
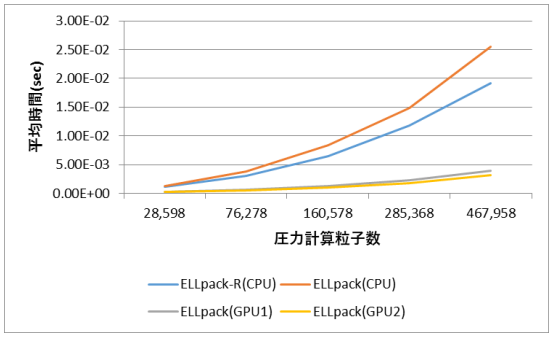


図3　3次元ダムブレイク問題

表1 CPU/GPUの計算結果



５．まとめ

本研究では3次元MPS法をダムブレイク問題に適用し，疎行列ベクトル積の性能について検証した．疎行列の格納方法にELLpack形式を適用し，CPUではELLpack形式とELLpack-R形式を適用し，計算規模を変化させ比較した．検証した結果，計算規模が大きいとGPUの高速化が顕著に表れた結果となった．しかしGPU計算においてはメモリバンド幅などまだピーク性能との差が見られることから，さらにチューニングの余地があると思われる．

「参考文献」

1) Nathan Bell and Michael Garland, "Efficient Sparse Matrix-Vector Multiplication on CUDA", *NVIDIA Technical Report NVR-2008-004*, December 2008

2) 久保田悠司,高橋大介「GPUにおける格納形式自動選択による疎行列ベクトルの高速化」情報処理学会研究報告，NO19(2010)，3.