

GPUを用いた個別要素法による粉体シミュレーション に対するメモリ使用量を抑えた高速化手法

渡辺, 勢也
東京工業大学総合理工学研究科

青木, 尊之
東京工業大学学術国際情報センター

都築, 怜理
東京工業大学総合理工学研究科

<https://hdl.handle.net/2324/7183267>

出版情報 : Transactions of the Japan Society for Computational Engineering and Science. 2016, pp.20160013-, 2016-05-27. 日本計算工学会

バージョン :

権利関係 : © 2016 The Japan Society For Computational Engineering and Science



GPUを用いた個別要素法による粉体シミュレーションに対する メモリ使用量を抑えた高速化手法

Performance Tuning on GPU for Granular Simulation based on Discrete Element Method Saving Memory Usage

渡辺勢也¹, 青木尊之², 都築怜理¹

Seiya WATANABE, Takayuki AOKI and Satori TSUZUKI

¹ 東京工業大学総合理工学研究科 (〒152-8550 東京都目黒区大岡山 2-12-1-i7-3)

² 東京工業大学学術国際情報センター (〒152-8550 東京都目黒区大岡山 2-12-1-i7-3)

For granular particle simulations based on Discrete Element Method (DEM), we have a severe problem of its long computational time consuming. Using GPU (Graphics Processing Unit) is one of options to accelerate the computation with the high performance of floating-point operation. Since the amount of on-board high-speed memory on GPU cards is limited to several GB, we have to choose algorithms to save the memory usage. Four kinds of speed-up techniques have been proposed: a highly efficient method for neighbor-particle searching, sorting the particle order on their positions, an efficient memory usage for the tangential spring, and fusion of GPU kernel function to reduce the memory access. A benchmark test of the 3-dimensional dam-breaking problem is examined to evaluate their performances and their memory usages for four techniques, respectively. The computational performance of the code which all the four techniques are applied to is improved 14.86 times higher than the original one, and only 6% increase of the memory usage is required. It is shown that the four speed-up techniques are quite available for GPU computing to achieve higher performance and less memory usage for DEM computation. We have also demonstrated a large-scale dam-breaking test using 15,728,640 particles on a NVIDIA Tesla K20X and the simulation has completed within 5.5 hours.

Key Words: GPU, Discrete Element Method, High-performance computing, Neighbor-particle searching method, particle sorting

1. 緒 言

レーザープリンターのトナーや、土砂崩れなどの斜面災害、錠剤の製造プロセスなどの様々な分野で、粉体現象が重要な役割をしている。粉体のシミュレーションには、個別要素法（別名、離散要素法）(DEM: Discrete Element Method または Distinct Element Method) ⁽¹⁾ が広く用いられている。DEM は粉体を有限の大きさを持つ粒子で表現し、周囲の接触している粒子からバネとダッシュポットによる力を受け、粒子が運動する計算モデルである。DEM で粉体の挙動をより精度良く計算するためには、計算に用いる粒子の大きさを現実の粉体と同程度にする必要がある。しかし、粉体の大

きさは計算対象のサイズに対して非常に小さいことが多く、しばしば数百万から数億個という膨大な数の粒子が必要となる。1 個の計算粒子が実際の数万個～数 100 万個の粉体粒子を代表する粗視化の手法もあるが、場合によっては現象の本質を失う危険性がある。また、DEM は陽的な時間積分を行うため、粒子を固い材質（実際のヤング率に近い）とした場合、計算の安定性を確保するために時間刻みを非常に小さくする必要がある。所望の時刻まで計算するには、1,000 万ステップを超えることも珍しくない。従って、DEM シミュレーションの計算コストは大きくなり、通常のスカラ・プロセッサの単一スレッドを用いた計算では、計算時間の制約から扱える粒子数は数十万個までに限定されてしまう。

計算の高速化が進めば、より大きな問題を解くことができる。一つの方法は複数プロセッサを使った並列計算である。DEM は接触相互作用であるため、粒子

* 原稿受付 2015 年 05 月 22 日, 改訂年月日 2015 年 09 月 16 日, 発行年月日 2016 年 05 月 27 日, ©2016 年 日本計算工学会.
Manuscript received, May 22, 2015; final revision, September 16, 2015; published, May 27, 2016. Copyright ©2016 by the Japan Society for Computational Engineering and Science.

番号などではなく計算空間の領域分割を行う必要がある．粉体粒子の空間分布は時間とともに変化するため、静的な領域分割では、領域内の粒子数に偏りが生じて各プロセッサの計算負荷のバランスが悪くなり、全体の計算性能が低下してしまう．そのため、分割した領域内にほぼ同じ数の粒子が含まれるように絶えず動的に領域を分割する必要がある^{(2)~(3)}．一方、画像表示用に設計されたGPU(Graphics Processing Unit)が高い浮動小数点演算性能とメモリバンド幅を持っているため、最近は演算アクセラレータとして多くのスパコンにも搭載されている．粒子計算にGPUを適用する研究^{(4)~(6)}、GPUを用いたDEMの高速化アルゴリズムの研究も行われている^{(7)~(10)}．

既往研究⁽¹⁰⁾のGPU計算における高速化手法はペアリスト法と呼ばれ、作用反作用の法則を用いて*i*番目の粒子と*j*番目の粒子間の相互作用の重複計算を行わないようにし、さらに1スレッドが1つの粒子ペア*i-j*間の接触力を計算している．粒子に加わる接触力の総和計算でメモリへの書き込み競合を回避するために、粒子ペアの数を要素数として持つ配列に各スレッドが計算した接触力を書き込み、その配列を参照して各粒子に加わる接触力の総和を計算する．これにより、当時のCPUに対して50倍程度高速なDEMのGPUプログラムを提案し、100万個の粒子を用いた計算を実行した．しかし、この手法は一時的に接触力を記憶する配列と、総和計算で作用反作用の法則を考慮するための参照表に多くのメモリが使用されてしまう．GPUのオンボードメモリ容量は数GBと制限があるため、粒子物理量以外に使用するメモリ量を極力抑える必要がある．

本研究では、GPUのオンボードメモリ容量が少ないことを考慮し、メモリ使用量を抑えた高速化手法を提案する．本研究で提案するDEMのGPUプログラムでは、作用反作用の関係を利用せず、GPU計算において1スレッドが1つの粒子を担当する．作用反作用の関係を用いないため、演算量は作用反作用の関係を用いた場合の2倍となるが、接触力の総和計算でメモリへの書き込み競合がおこらないため、総和計算のための配列が不要になり、既往研究⁽¹⁰⁾よりもメモリ使用量を大幅に削減できる．高速化手法として、効率のよい近傍粒子探索手法、粒子物理量のソートによるキャッシュヒット率の向上、接線方向の接触力の計算手法、Kernel Fusionによるメモリアクセス回数の低減を提案する．3次元のダム崩壊問題で計算性能とメモリ使用量を評価し、それぞれの高速化手法の効果を示す．

2. DEM(Discrete Element Method)

DEMは粉体を構成している個々の粒子の運動を時間積分することにより全体の挙動をシミュレーションする計算手法である．重力などの外力と粒子間に働く力の合力に対して、ある時点における粒子の位置と速度を時間積分することにより、1タイムステップ後の粒子

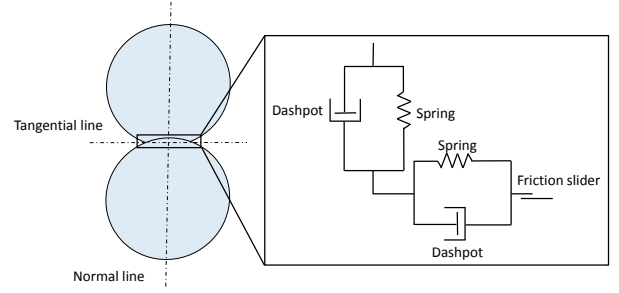


Fig.1 DEM interaction model.

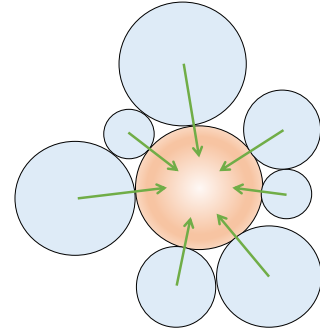


Fig.2 Collisions among neighbor particles.

の位置と速度に更新する．粒子間の相互作用は、Fig.1に示すように、粒子同士が接触している場合のみ働くバネとダッシュポットにより表現される．DEMの計算モデルでは、バネは粒子同士の食い込み深さに比例した反発力、ダッシュポットは粒子間の相対速度に比例した減速力を粒子に与える．粒子に加わる力は法線方向に働く反発力と、接線方向に働く摩擦力に分解され、接線方向のスライダは粒子の摩擦係数に応じて接線方向の力に上限を与える．*i*番目の粒子が*j*番目の粒子から受ける法線方向および接線方向の力は以下の式によって表される．

$$\mathbf{F}_{ij}^N = k^N \mathbf{L}_{ij}^N + c^N \frac{\Delta \mathbf{L}_{ij}^N}{\Delta t} \quad (1)$$

$$\mathbf{F}_{ij}^T = k^T \mathbf{L}_{ij}^T + c^T \frac{\Delta \mathbf{L}_{ij}^T}{\Delta t} \quad (2)$$

ここで、 $\mathbf{L}$ はバネの圧縮量ベクトル、 $\Delta \mathbf{L}$ は相対変位増分ベクトル、 Δt は時間刻み、 k はバネの弾性係数、 c はダッシュポットにおける粘性減衰係数である．添字NとTはそれぞれ法線方向と接線方向を表す．接線方向のバネの圧縮量ベクトルは、現在の時間ステップにおける粒子の位置や速度の関係から求めることはできない．そのため、粒子同士が衝突してから現在までの各ステップの相対変位増分ベクトルを足し合わせていくことで、現在の時刻の接線方向のバネの圧縮量ベクトルを求める．粒子同士が離れた場合は、バネの圧縮量ベクトルを開放する．粒子*i*の半径を R_i とすると、接線方向の力による粒子*i*に加わるモーメントのベク

トル $\mathbf{M}_{ij}$ は次のように表される．

$$\mathbf{M}_{ij} = \frac{R_i}{|\mathbf{x}_j - \mathbf{x}_i|} (\mathbf{x}_j - \mathbf{x}_i) \times \mathbf{F}_{ij}^T \quad (3)$$

Fig.2 のように粉体の一つの粒子は一度に複数の周囲の粒子と接触する可能性があり，粒子 i と接触しているすべての粒子 j に関して力とモーメントのベクトルを計算し，粒子 i に作用する力とモーメントの合力ベクトルを求める必要がある．粒子の並進および回転を中心とする回転の運動方程式は以下の式で表される．

$$m_i \frac{d^2 \mathbf{x}_i}{dt^2} = \sum_{i \neq j} \mathbf{F}_{ij} + m_i \mathbf{g} \quad (4)$$

$$I_i \frac{d^2 \boldsymbol{\theta}_i}{dt^2} = \sum_{i \neq j} \mathbf{M}_{ij} \quad (5)$$

ここで， $\mathbf{x}_i$ と $\boldsymbol{\theta}_i$ は粒子 i の位置および回転角ベクトル， m_i と I_i は粒子の質量および慣性モーメントである．各粒子に対して並進および回転の運動方程式を数値積分することで，新しい時刻における粒子の位置，速度，回転角，角速度を求める．

3. GPU による DEM の高速化手法

3.1 リンクリストと粒子登録法による近傍粒子探索 DEM では接触している粒子間のみ相互作用が働くので，接触しないことが明かな遠く離れた粒子との接触判定は無駄である．計算領域を均一な格子（セル）で分割し，自身の所属するセルおよび周囲のセル内の粒子とのみ接触判定を行うことで，相互作用計算の高速化が行える⁽¹¹⁾．全ての粒子が同じ大きさの場合は，格子の大きさを粒子直径（相互作用半径）と同じにする．近傍探索に用いるメモリ使用量を削減するために，各セルが所属する粒子番号を全て記憶するのではなく，Fig.3 のように各粒子が同一セル内の他の粒子を数珠つなぎに記憶するリンクリスト⁽¹²⁾ がしばしば用いられている．

自身の所属するセルおよび周囲のセル内には，着目している粒子と接触しない粒子も多く含まれる．そこで，Fig.3 のように粒子登録法⁽¹³⁾ により，周囲のセルからさらに近傍の粒子を絞り込むことで，近傍探索を効率よく行うことができる．リンクリストと粒子登録法を組み合わせた近傍探索手法は，流体計算のための粒子法である SPH⁽¹⁴⁾ の CPU 計算で行われているが⁽¹⁵⁾，DEM には適用例が報告されていない．本論文では DEM にこれらを適用した GPU コードを実装し有効性を検証する．各粒子が近傍粒子リストとして式 (6) の半径 R_c (Fig.3 の黄色の実線の円) 内に含まれる粒子を記憶し，ある一定時間の間は近傍粒子リスト内の粒子とのみ接触判定を行う． rc は粒子の相互作用半径 (Fig.3 の黄色の点線の円) である．

$$R_c = rc + \alpha D_{\min} \quad (6)$$

ここで， $D_{\min}$ は最小の粒子直径， α は近傍粒子リストの更新頻度を調整するパラメータである．自身の所属

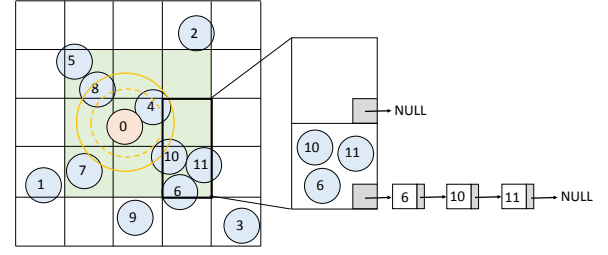


Fig. 3 Neighbor-particle searching by means of book-keeping method with linked-list.

するセルおよび周囲のセル内の粒子との距離計算のみで近傍粒子リストを作成するために，セルの大きさは R_c と等しく設定する．パラメータ α を小さくすると，近傍粒子リストを作成する半径が小さくなり，相互作用計算する粒子が少なくなるため，相互作用の計算時間は短縮される．しかし，近傍粒子リストの更新回数が多くなり，近傍粒子リストの作成にかかる時間が増えるため，計算全体としては計算時間が短くならない場合もある．近傍粒子リストは毎ステップ更新されるのではなく，近傍粒子リストに含まれていない粒子が注目している粒子の影響半径 rc 内に入った場合に更新を行う．実際の実装では，1 タイムステップでの各粒子の移動距離を，各粒子が保持するリストが更新されてからの粒子移動距離 x_{book} に毎ステップ足し合わせる． x_{book} が $(R_c - rc)/2$ を上回る粒子が一つでもあった場合，全粒子の近傍リストの更新を行う．このとき，粒子移動距離の積載値 x_{book} を初期化する．

3.2 粒子物理量のソート 位置や速度などの粒子の物理量を式 (7) で計算される各粒子のセル番号の順に並び替えることで，相互作用計算で接触粒子の物理量へのメモリ参照が GPU のデバイスメモリに対するコアレスアクセスとなり，キャッシュにも乗りやすくなる⁽⁸⁾．既往研究⁽⁸⁾ では，粒子番号のソートを前提とした近傍探索手法を用いられているが，本論文ではリンクリストと粒子登録法による近傍探索手法に粒子物理量のソートを実装し有効性を検証する．

$$cell_{id} = \text{int} \left(\frac{x}{\Delta x} \right) + \text{int} \left(\frac{y}{\Delta y} \right) \times nx + \text{int} \left(\frac{z}{\Delta z} \right) \times nx \times ny \quad (7)$$

ここで， x, y, z は粒子の座標， $\Delta x, \Delta y, \Delta z$ は x, y, z 軸方向のセルの大きさ， nx, ny は x, y 軸方向のセルの数である．ただ，DEM では各粒子は位置や速度の他にも，接触している粒子との接線方向のバネの圧縮量を変数として持つため，一つの粒子に対してソートを行う変数が多く，粒子の物理量のソートに時間がかかる．そのため，毎ステップソートを行うとソートの時間が相互作用計算などよりも長くなり，計算全体の高速化が行えなくなる．DEM の時間刻み幅は小さいため，1 タイムステップ間での粒子の移動距離は小さく，

粒子分布はほとんど変化しない．そこで，ソートの頻度を数百ステップに一度行うようにすることでソートのオーバーヘッドを小さくし，全体の計算時間を短縮できる．本研究では，粒子登録法の近傍粒子リストは数十ステップに一度更新されるため，リストが30回更新された時に粒子の再整列を行うようにすることで，数百ステップに一度再整列されるほぼ最適な頻度を設定している．

3.3 接線方向のバネ圧縮量の取り扱い DEMでは接線方向のバネの圧縮量を現在の時刻の粒子の位置関係から求めることはできないので，粒子が接触した時刻から各時刻におけるバネ圧縮量の増分を蓄積しておく必要がある．各粒子は，全粒子とのバネの圧縮量を変数として持つのではなく，接触する最大の粒子数（3次元計算ですべての粒子の大きさが等しい場合は12個）のバネの圧縮量を保持する変数を持つ．Fig.4のように，バネの圧縮量に対応させた接触粒子リストに，接触している粒子の番号を記憶させる．着目する*i*番目の粒子と周囲の*j*番目の粒子との相互作用計算では，*i*番目の粒子の接触粒子リストに*j*番目の粒子が登録されているかを調べ，前の時刻で接触していたかを判定する．前の時刻で接触していた場合は対応したバネ圧縮量を参照し，接触していない場合は接触粒子リストの空いている要素を*j*番目の粒子に割り当てる．接触していた粒子が離れたときは，接触粒子リストの粒子番号とバネの圧縮量の値を開放する．しかし，この方法では，接触粒子リストに登録されているかの判定，粒子が離れたときの接触粒子リストからの粒子番号の削除のときに接触粒子リストをたどる必要があり，メモリアクセス回数が多くなる．

本研究では，接触粒子リストと粒子登録法の近傍粒子リストを統合することで，接触粒子リストをたどることなくバネの圧縮量を参照できる手法を提案する．Fig.5に示すように，接触していない近傍の粒子も接触粒子リストに加え，接触していない粒子のバネ圧縮量は0にする．粒子登録法の近傍粒子リストの更新のときに，新しい近傍粒子リストに対応させるように接触粒子リストとバネの圧縮量のソートを行う．相互作用計算では，近傍粒子リストの先頭の粒子から計算していくので，バネの圧縮量は先頭から順々に参照すればよい．粒子が離れた場合のバネの圧縮量の開放は，バネの圧縮量配列の対応する要素を0にするだけで，接触粒子リストには粒子番号を登録したままとなる．

接触粒子リスト (*ContactList*) と接線方向のバネの圧縮量は，要素数が (全粒子数 *NP*) × (最大の接触粒子数 *NCP*) の1次元配列で管理する．接触粒子リストをたどる実装では，各粒子の接触粒子リストの要素が連続するよう粒子番号とバネの圧縮量を格納していたが，各粒子の接触粒子リストの*k*番目のメモリの並びが連続になるように変更する．バネの圧縮量の配列へのアクセスは式(8)から式(9)に変わる．

$$ContactList[i \times NCP + k] \quad (8)$$

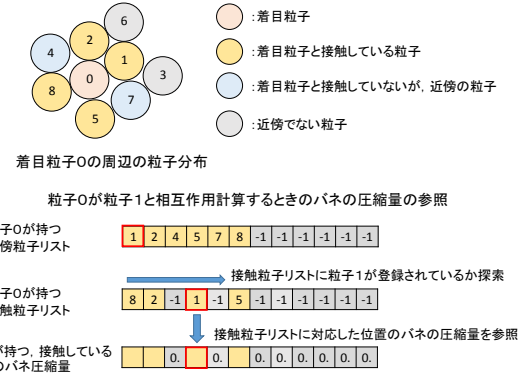


Fig.4 Reference of spring compression by searching.

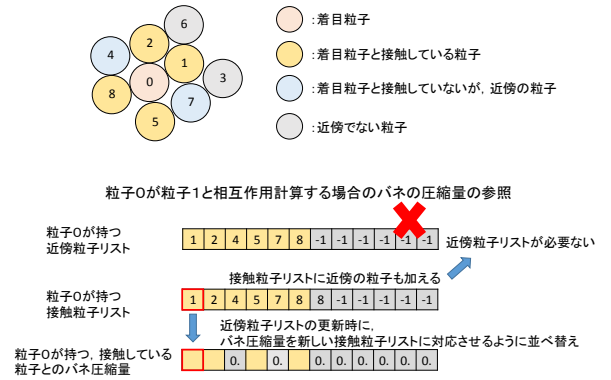


Fig.5 Reference of spring compression by sorting list.

$$ContactList[i + k \times NP] \quad (9)$$

ここで，*i* は着目粒子番号，*i* 番目の粒子の接触粒子リストの*k* 番目の要素，*NP* は全粒子数，*NCP* は最大の接触粒子数である．また，近傍粒子リストは接触粒子リストとバネの圧縮量のソートのために一時的にレジスタに作成するだけでよく，粒子登録法に必要なメモリ量を大幅に削減することができる．

3.4 Kernel Fusion によるメモリアクセスの削減 Fig.6 に GPU 実装した DEM 計算のフローチャートを示す．上から順に，粒子物理量のソート，各セルのリンクリストの作成，粒子登録法の近傍粒子リストの作成，接触判定と粒子間相互作用の計算，粒子の時間積分，粒子登録法の近傍粒子リストの更新の判定（粒子移動距離の積載値 x_{book} の計算）の6つのサブルーチンで構成される．毎ステップ必ず行うのは，フローチャートの赤枠で囲んだ，粒子間相互作用の計算，粒子の時間積分，近傍粒子リストの更新の判定である．この3つの計算を行うカーネル関数を1つのカーネル関数にまとめる (Kernal Fusion) ことにより，メモリアクセス回数を削減して高速化を図る．

Table 1 Calculation condition.

Case	Implemented tuning	Neighbor-particle searching method	Book-keeping parameter α
3-0	—	Linked-list	—
3-1	Section 3-1	Book-keeping with linked-list	0.05
3-2	Section 3-1, 3-2	Book-keeping with linked-list	0.05
3-3	Section 3-1, 3-2, 3-3	Book-keeping with linked-list	0.08
3-4	Section 3-1, 3-2, 3-3, 3-4	Book-keeping with linked-list	0.08

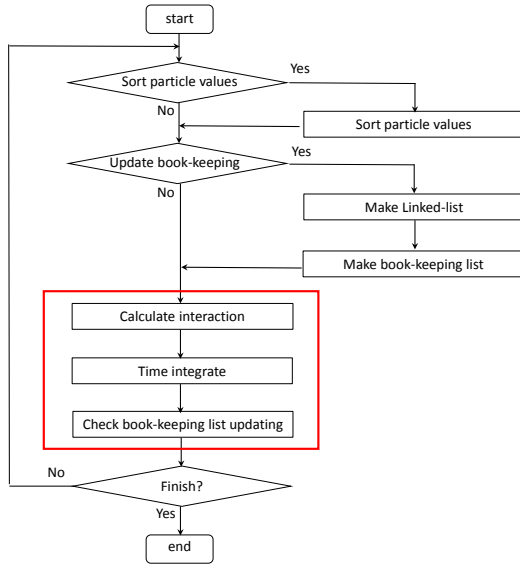


Fig.6 Flow chart of DEM simulation.

それぞれ別のカーネル関数で行う場合は、相互作用計算で得られた各粒子に加わる力をグローバルメモリに書き出し、時間積分を計算するときに力を読み込む必要がある。カーネルをまとめることで相互作用計算で得られた力をレジスタに保持し、時間積分のときはレジスタから力を読み込む事ができる。

4. 実行性能評価に用いる例題

3章で提案した高速化手法の評価を行うために、3次元のダム崩壊問題のシミュレーションを131,072個の粒子を用いて行った。詳しい計算条件と物理条件をTable 1とTable 2に記す。本研究で提案した高速化手法の効果を調べるため、メモリ使用量が少ない高速化手法であるリンクリストを用いたセル分割法^{(2)~(3)}のみで近傍探索を行った場合(case3-0)を基準とする。3.1節から3.4節で示した高速化手法を順に追加していき(case3-1からcase3-4)、各節で示した手法の相対的な効果も調べる。粒子登録法の近傍粒子リストの更新頻度を設定するパラメータ α は、パラメータチューニングにより設定し、最も計算時間が短かったものを採用している。

計算領域の隅にダム崩壊前の初期粒子配置をFig.7のように設定する。この初期配置は、粒子を格子点状

Table 2 Physical condition.

Particle radius	1.0×10^{-3} m
Particle mass	1.0×10^{-5} kg
Spring constant (Normal)	4.0×10^3 N/m
Spring constant (Tangential)	1.6×10^3 N/m
Damping coefficient (Normal)	0.4 Ns/m
Damping coefficient (Tangential)	0.25 Ns/m
Coefficient of friction	0.3
Discrete time	5.0×10^{-6} sec

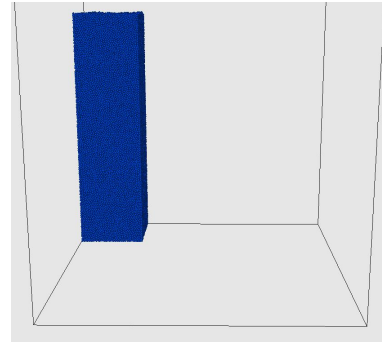


Fig.7 Initial condition of the breaking dam.

に均等に並べたあと、そこから擬似乱数を用いてわずかに各粒子をずらして配置し、自由落下させて安定になるまで計算した状態である。Fig.8は、初期状態から10万ステップの計算を行ったダム崩壊後の状態ある。高速化手法の性能比較では、Fig.7の初期配置からFig.8までの10万ステップの計算を行うのに要した時間を測定する。粒子の物理量は単精度の浮動小数点数であり、時間積分にはLeap-frog法を用いている。

計算の実行環境には、東京工業大学学術国際情報センターのTSUBAME2.5を用いた。使用したGPUはNVIDIA社のTesla K20Xであり、CUDAのVersionは6.5である。

5. 実行性能評価の結果

5.1 提案手法の計算時間測定 Table 3に各条件での計算時間の内訳を示す。左から順に、粒子物理量

Table 3 Computation time of each case.

Case	Sort	Linked-list	Book-keeping	Interaction	Time integral	Checking list update	Total
3-0	—	16.85	—	899.71	13.66	—	934.13
3-1	—	1.36	19.24	281.20	13.63	5.12	324.62
3-2	2.13	1.16	9.65	249.75	13.54	5.15	285.39
3-3	0.64	0.62	6.65	49.67	12.04	4.91	78.58
3-4	0.64	0.62	6.68		52.64		62.85

unit:sec

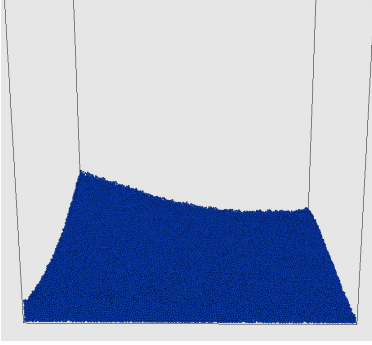


Fig.8 A breaking dam simulation (step = 100,000).

のソート (Sort), リンクリストの作成 (Linked-list), 粒子登録法の近傍粒子リストの作成 (Book-keeping), 接触判定と粒子間相互作用の計算 (Interaction), 時間積分 (Time integral), 粒子登録法の近傍粒子リストの更新の判定 (Checking list update), 全体の計算に要した時間 (Total) を秒で示している. Table 4 に高速化の効果を, 1 つ前の case との相対的な比較 (Relative speed-up) と高速化していない実装 (case3-0) との比較 (Total speed-up) で示している.

case3-1 では, 近傍粒子探索で周囲のセルから近傍の粒子を絞り込み接触判定を行うことで, 接触していない粒子との無駄な接触判定が減り, 接触判定と相互作用計算を行うカーネル関数の実行時間を 31.3% に削減できた. 近傍粒子リストの作成は大きなオーバーヘッドとならず, 計算全体で 2.88 倍の大幅な高速化を達成した.

case3-2 で行った粒子物理量の再整列の効果は, 粒子登録法の近傍リストの作成時間を 50% に削減, 粒子間相互作用カーネルの実行時間を 88% に削減でき, 計算全体で 1.14 倍程度に高速化した. 粒子の再整列の頻度を調整したことにより, 再整列にかかる時間は計算全体の 1% 以下となり, 粒子再整列のオーバーヘッドは殆ど無視できる. ソートの回数を増やしても粒子登録法の近傍粒子リストの作成と相互作用計算の時間はあまり短縮されず, ソートのオーバーヘッドにより計算全体の時間が延びる結果となった.

case3-3 で実装した接線方向のバネの圧縮量の取り扱い方法は相互作用計算の時間を 20% に短縮し, 極めて

効果が高かった. これは接線方向のバネの圧縮量を参照するときのメモリアクセス回数を大幅に減らすことができたこと, 接触リストとバネの圧縮量のデータ構造を変えて, バネ圧縮量の配列へのアクセスがコアレスアクセスになったためである. 粒子登録法の近傍リスト作成時にバネの圧縮量のソートが加わるため, 近傍リストを作成する時間が長くなったが, リストを作成する半径を大きく取り更新回数を減らしたため, 結果的には近傍リストの作成時間は短くなった. 近傍リストの更新回数が少なくなったことにより, 粒子の再整列とリンクリストの作成回数も少なくなり, それぞれの計算時間も短縮された. 接線方向のバネの圧縮量の取り扱い方を変更することで, 計算全体で 3.63 倍の大幅な高速化が行えた.

case3-4 では, 粒子間相互作用, 粒子の時間積分, 粒子登録法の近傍リストの更新の判定を行う 3 つのカーネル関数を, 1 つのカーネル関数にまとめてメモリアクセス回数を減らした. case3-3 では 3 つのカーネル関数の実行時間の合計は 66.62 秒であったが, 1 つのカーネルにまとめることで 52.64 秒に短縮することができた. 計算全体では 1.25 倍の高速化が行えた.

高速化手法を行っていない case3-0 と比較すると, 提案した 4 つの高速化手法を全て実装することで, 計算全体で 14.86 倍の大幅な高速化が行えた.

既往研究との実行性能の比較のために, 1 秒間に 1 タイムステップ更新できる粒子数 (粒子数/1 タイムステップの計算時間) を求めた. 最も計算の速い case3-4 では, 131,072 粒子, 100,000 ステップの計算を 62.85 秒で完了しているので, 実行性能は $2.0 \times 10^8 \text{sec}^{-1}$ である. ペアリスト法を用いた既往研究⁽¹⁰⁾の実行性能は NVIDIA Tesla C1060 を用いて $5.0 \times 10^7 \text{sec}^{-1}$, 複数 GPU で計算を行った既往研究⁽³⁾では, NVIDIA Tesla K20X を 32 台利用して $2.0 \times 10^7 \text{sec}^{-1}$ であり, 本論文の提案手法は既存手法よりも高速に計算を実行している.

複数 GPU による計算では, GPU 間の通信がホスト側の CPU を介して行われるため通信に時間が掛かる. 提案手法は演算部分を大幅に高速化することができたが, 複数 GPU で計算を行った場合, GPU 間の通信時間の占める割合が既往研究^{(2)~(3)}よりも増えるため, GPU 数に対して線形に実行性能は増加しないと予想

Table 4 Speed-up of each case.

Case	Relative speed-up	Total speed-up
3-0	1.00	1.00
3-1	2.88	2.88
3-2	1.14	3.27
3-3	3.63	11.89
3-4	1.25	14.86

される。

5.2 メモリ使用量の比較 各提案手法で性能評価を行ったときに確保したメモリ量を Fig.9 に示す。縦軸にメモリ使用量を MB で示し、青色が位置や速度などの粒子物理量 (particle variables), 灰色がリンクリスト法 (linked-list), 黄色が粒子登録法 (book-keeping), オレンジ色が粒子再整理 (sort) に必要なメモリ量を表している。

case3-1 で実装した粒子登録法のメモリ使用量は, 131,072 個の粒子を用いた場合には 7 MB であり, リンクリスト法よりは少ないが全体の 13% を占めている。計算領域を分割する格子のサイズを, 影響半径 rc から近傍リストを作成する半径 Rc に変えたため, 格子の数が少なくなり, リンクリスト法のメモリ使用量を 10.1 MB から 8.6 MB に減らすことができています。全体では, 5.5 MB の増加である。

case3-2 では, 粒子物理量のソートのための配列を確保しているため, メモリ使用量が 2.5 MB 増加している。

case3-3 では, 接線方向のバネの圧縮量を管理する接触粒子リストに, 粒子登録法の近傍粒子リストを組み込んだことで, 粒子登録法のメモリ使用量を 7.0 MB から 1.0 MB に削減できている。

case3-4 では, Kernel Fusion を行ったことで粒子に作用する力とモーメントの配列は必要なくなったが, 更新前後の粒子の位置, 速度, 角速度の配列を持つ必要があるため, 粒子の物理量が 1.5 MB 増えている。カーネルが分かれているときは, すべての粒子の相互作用計算が終わった後に粒子の時間発展を行っていた。しかし, カーネルを一つにまとめた場合は, 他の粒子が相互作用を計算する前に時間積分が行われる粒子が存在するため, 時間積分後の物理量の配列が必要になる。

高速化を行っていない case3-0 のメモリ使用量の合計は 49.1 MB であるのに対し, 14.86 倍の高速化を達成した case3-4 は 52.0 MB で, わずか 6% のメモリ使用量の増加である。すなわち, 本研究で提案した高速化手法は, メモリ消費を抑えて高速化が実現できたといえる。既往研究⁽¹⁰⁾ のペアリストを用いる方法では, セルを分割することによる近傍探索に加えて, GPU でスレッド並列化するためのペアリストや一時的に接触力を記憶する配列, 作用反作用の法則を適用するため

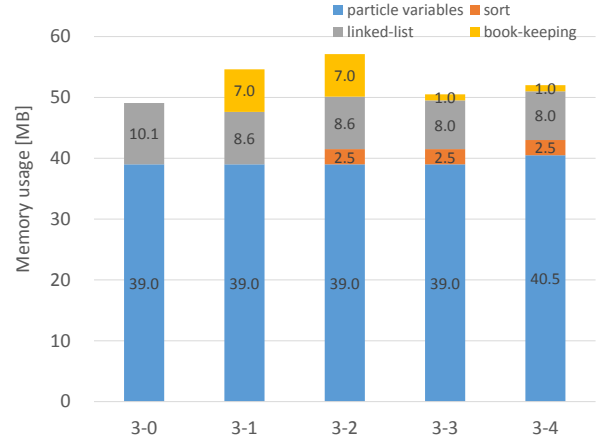


Fig.9 Memory usage of each case.

の力の参照表などが必要となる。近傍粒子のリストや参照表は (粒子数) $\times$ (近傍の粒子数の最大値) を要素数とする配列であり, ペアリストや一時的に接触力を記憶する配列は粒子ペアの数を要素数とする配列である。実際には粒子ペアの数は粒子数よりも遥かに多く, 簡単のために粒子ペア数が粒子数の 3 倍と仮定して, 粒子物理量以外的高速化に必要なメモリ使用量を見積もると, 本論文の手法の 3 倍以上になる。

計算領域の大きさや粒子数を変化させた場合のメモリ使用量は, 粒子物理量と粒子再整理, 粒子登録法は粒子数に対して線形に増加する。リンクリスト法は, 粒子数と計算領域の大きさ (分割する格子の数) に比例して増加する。

6. 提案手法による大規模計算

本研究で提案した高速化の有効性を示すために, GPU NVIDIA Tesla K20X のオンボードメモリ 6 GB で確保できる最大の数の粒子を用いて, 大規模なダム崩壊問題のシミュレーションを行った。計算の物理条件は Table 2 と同じであり, 粒子数は 15,728,640 である。Fig.10 にダム崩壊途中の様子を示す。粒子の色は, 粒子の速さが大きいほど赤く, 小さいほど青くしている。1 秒間に 10 タイムステップの計算が行えていて, ダムが崩壊するまでの 20 万ステップの計算を約 5 時間半で完了している。これまでは計算時間の理由で複数の GPU を用いて行われてきた粒子数 1000 万以上の大規模な DEM 計算^{(2)~(3)} が, 本研究で提案した高速化手法により単一 GPU で実現できた。メモリ使用量は全体で 5520 MB で, 内訳は粒子物理量に 4860 MB, リンクリスト法に 240 MB, 粒子登録法に 120 MB, 粒子再整理に 300 MB である。既存手法であるペアリスト法で 6GB のメモリを利用した場合は, 本論文の手法に比べて扱える粒子数が 500 万個ほど少なくなると考えられる。

7. 結言

GPU を用いて DEM による粉体シミュレーションを

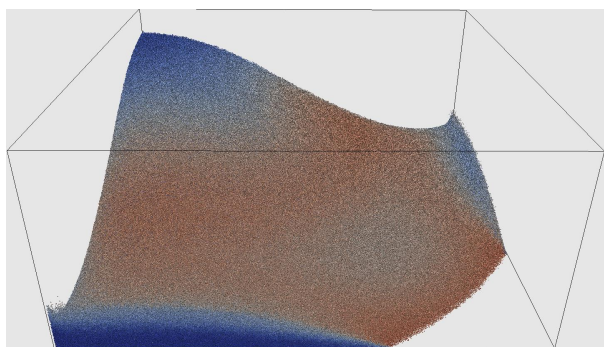


Fig. 10 A large-scale simulation using 16 million particles on 1GPU.

高速化する際、GPU のオンボードメモリが数 GB と少ないことを考慮し、メモリ使用量を抑えた高速化手法を提案し、高速化手法を用いていない GPU プログラムに対して計算時間とメモリ使用量を比較した。

高速化手法として、リンクリストと粒子登録法を用いた近傍探索手法、粒子物理量のソートによるメモリアクセスの改善、接線方向のバネの圧縮量の効率的なアクセスの方法、毎ステップ実行されるカーネル関数を一つにまとめる (Kernal Fusion) ことによるメモリアクセスの削減手法を提案した。特に効果の高かった手法は、リンクリストと粒子登録法による近傍探索手法と、バネの圧縮量の参照方法の改善であり、実装することによりそれぞれ 2.88 倍、3.63 倍の高速化を行えた。提案手法を全て実装することで、高速化を行っていない GPU プログラムに比べて 14.86 倍の高速化を実現した。メモリ使用量はわずか 6 % の増加であり、本研究で提案した高速化手法は、メモリ使用量の増加を必要最小限に抑えて高速化を実現できたといえる。

提案した高速化手法を実装した GPU プログラムで、NVIDIA Tesla K20X のオンボードメモリ容量 6GB を最大限利用し、15,728,640 個の粒子を用いた大規模な 3 次元ダム崩壊問題を行った。従来、粒子数が 1,000 万個を超える規模の DEM 計算は計算時間がかかり過ぎるため単一 GPU では実現されていなかった。本研究で提案した高速化手法により、スパコンで行われている大規模 DEM 計算が通常のワークステーションで実現可能になる。また、本研究の高速化手法の一部は、DEM 以外にも近接相互作用に基づく SPH 法や MPS 法⁽¹⁶⁾にも適用でき、その有用性は高い。

謝辞 研究の一部は、科学研究費補助金・基盤研究 (S) 課題番号 26220002「ものづくり HPC アプリケーションのエクサスケールへの進化」、科学技術振興機構 CREST「ポストペタスケール高性能計算に資するシステムソフトウェア技術の創出」、学際大規模情報基盤共同利用・共同研究拠点、および革新的ハイパフォーマンス・コンピューティング・インフラから支援を頂いた。記して謝意を表す。

参考文献

- (1) Cundall, P. A. and Strack, O. D., A discrete numerical model for granular assemblies, *Geotechnique*, **29**, pp. 47–65, 1979.
- (2) 都築怜理, 青木尊之, GPU による大規模粒子法シミュレーションの実問題への適用, 日本計算工学会第 19 回計算工学講演会講演論文集, 2014.
- (3) Tsuzuki, S. and Aoki, T., Large-scale granular simulations using Dynamic load balance on a GPU supercomputer, Proceedings of the 2014 ACM/IEEE conference on Supercomputing (SC'14), 2014.
- (4) Rustico, E., Bilotta, G., Gallo, G., Herault, A., Del Negro, C., and Dalrymple, R. A., A journey from single-GPU to optimized multi-GPU SPH with CUDA, 7th SPHERIC Workshop, 2012.
- (5) Liu, F., Harada, T., Lee, Y., and Kim, Y. J., Real-time collision culling of a million bodies on graphics processing units, *In ACM Transactions on Graphics (TOG)*, **29**, No. 6, pp. 154, 2010.
- (6) Govender, N., Wilke, D. N., Kok, S. and Els, R., Development of a convex polyhedral discrete element simulation framework for NVIDIA Kepler based GPUs, *Journal of Computational and Applied Mathematics*, **270**, pp. 386–400, 2014.
- (7) 渡悠介, 酒井幹夫, 越塚誠一, 山田祥徳, GPU による離散要素法シミュレーションの高速化, 粉体工学会誌, **45**, No. 11, pp. 758–765, 2008.
- (8) Green, S., Particle simulation using cuda, *NVIDIA Whitepaper*, 2010.
- (9) 西浦泰介, 阪口秀, 粒子径分布を有する粉粒体に対する DEM シミュレーションの GPU 高速化手法, 計算工学会論文集, **2010**, Paper No. 20100008, 2010.
- (10) 西浦泰介, 阪口秀, GPU を用いた DEM の高速化アルゴリズム, 計算工学会論文集, **2010**, Paper No. 20100007, 2010.
- (11) Grest, G. S., Dunweg, B. and Kremer, K., Vectorized link cell Fortran code for molecular dynamics simulations for a large number of particles, *Computer Physics Communications*, **55**, No. 3, pp. 269–285, 1989.
- (12) 平林久義, 佐藤雅弘, 線形リストを用いた粒子法の近傍粒子探索, 計算工学会論文集, **2010**, Paper No. 20100001, 2010.
- (13) 小口かなえ, 澁田靖, 鈴木俊夫, GPU を用いた分子動力学法解析の高速化, 日本金属学会誌, **76**, No. 7, pp. 462–467, 2012.
- (14) Monaghan, J. J., An introduction to SPH, *Computer physics Communications*, **48**, No. 1, pp. 89–96, 1988.
- (15) Viccione, G. and Bovolín, V. and Carratelli, E. Pugliese, Defining and optimizing algorithms for neighbouring particle identification in SPH fluid simulations, *International Journal for Numerical Methods in Fluids*, **58**, No. 6, pp. 625–638, 2008.

- (16) Koshizuka, S. and Oka, Y., Moving-particle semi-implicit method for fragmentation of incompressible fluid, *Nuclear science and engineering*, **123**, No. 3, pp. 421–434, 1996.