

重心ボロノイ分割を用いた 並列粒子法のための動的負荷分散法

河野 瑛^{†1} 田浦 健次朗^{†1}

本稿では、並列粒子法において動的に負荷分散を是正する手法を提案する。提案手法では、重心ボロノイ分割を用いて空間情報に対応したデータの分割を行い、さらに物理的に近傍のデータを持つプロセッサの負荷情報をもとに領域の拡大・縮小を行うことで、動的な負荷分散の是正を可能にする。評価の結果、提案手法によって、優れた負荷分散が可能であることを確認した。また、MPIによる実装を行い、分散メモリ環境において128プロセスによる計算で、約52倍の性能向上を得た。

Dynamic Load Balancing Algorithm for Parallel Particle Method with Centrodial Voronoi Tessellation

AKIRA KONO^{†1} and KENJIRO TAURA^{†1}

In this paper we present a dynamic load balancing for parallel particle method simulation. The solution method proposed, distributes particle data using CVT (Centrodial Voronoi Tessellation). It balances the load to all the processors by expanding or shrinking the area, which a single processor is in charge of. This is done using load information based on the number of particles at one processor.

Evaluation demonstrated the effectiveness of our load balancing method, and got around 52 speed-ups at 128 processor calculation in distributed memory with MPI.

1. 序 論

近年、数値計算によるシミュレーションが様々な分野で行われており、並列化による高速化も重要なトピックとなっている。粒子法はこのような数値計算手法の一つであり、主に流体解析や構造解析に用いられている。比較的新しい手法であり、他の手法に比べて、計算格子を作成する必要が無い、大変形や破壊に適用しやすい、といったメリットを持つために、現在までに自由表面を有する流れや、剛体と流体の練成問題、布のシミュレーションなどの、流体や固体の複雑な動きを扱うモデルが研究されている。

また、粒子法においても大規模計算への要望は強く、GPUを用いて400万粒子による流体のダム崩壊問題を解いた例²⁾や、共有メモリ環境で24プロセッサを用いて1200万粒子による川のシミュレーションを行った例³⁾、スーパーコンピュータを用いて5500万粒子による剛体の流体への落下のシミュレーション⁶⁾を行った例など、様々な研究が報告されている。

粒子法では、同期的に計算が進むため、並列計算における各タイムステップの計算時間は最も遅いプロセッサが律速する。従って、粒子法を効率的に並列計算するためには、プロセッサ間の負荷バランスをできるだけ均等に保つことが重要である。しかしながら、粒子法の扱う計算モデルでは、計算領域に対して粒子分布が非一様であったり、粒子が激しい動きを伴うことが多いため、均等な負荷バランスを保つことは容易ではない。

本研究では粒子法を高速に計算するために、並列実行において効率的に負荷分散を行うことのできる手法を提案する。特に、計算領域に対して粒子分布が非一様であったり、時間経過とともに激しく変化したりするような、一般に並列計算において負荷分散が難しいモデルに対して、領域を低オーバーヘッドに変化させることで、動的に負荷バランスの均衡を保つことのできる手法を目指す。

2. 粒 子 法

本章では、本研究の想定する計算法である粒子法について、概要と性質、並列性能を引き出すための鍵となる点を述べる。

2.1 概 要

粒子法は、流体解析や構造解析などに用いられる連続体の数値計算手法のひとつであり、連続体を仮想的な粒子の集合と捉え、粒子上に物理量を配置することで離散化を行う。粒子

^{†1} 東京大学大学院 情報理工学系研究科

Graduate School of Information Science and Technology, The University of Tokyo

法において、空間上の任意の点 r における物理量 $A(\vec{r})$ は、式 (1) によって計算される。

$$A(\vec{r}) = \int A(\vec{r}') W(\vec{r} - \vec{r}', h) d\vec{r}' \quad (1)$$

式 (1) において、 W はカーネルと呼ばれる重み付け関数であり、距離が近いほど大きく、遠いほど小さくなり、 h で表されるカットオフ距離以降はゼロになるような関数が使われる。

従って、粒子法では距離が h 以下であるような粒子同士のみが、力を及ぼし合うことになる。

この空間離散化に従って各粒子上の物理量と力を計算し、速度や位置を更新しながら時間発展していく。

2.2 計算性質と効率的な並列化の鍵

2.2.1 負荷分散

粒子法のような、タイムステップ毎に同期して反復的に計算を行う手法では、各タイムステップの計算時間は最も遅いプロセスに律速される。従って、負荷を均等に分散することが高速化のための鍵となる。粒子法において、計算の計算時間はおよそ粒子数に比例すると考えられるので、各プロセスの担当する粒子数を均等に分配することが重要である。

2.2.2 局所性

ある粒子は物理的に近傍に存在する粒子としか相互作用しないことから、計算には空間的な局所性が存在する。従ってこの空間上の粒子配置と、実際に計算する際のメモリ上への粒子データ配置を対応させることによって、計算を効率的に行うことができる。従ってこの局所性から、並列計算を行う際には領域分割によるデータの分配が効果的である。

このとき、あるプロセッサについて、自分の担当する領域内の粒子を内点と呼び、他プロセッサの領域に存在し、内点と相互作用を行う粒子を外点と呼ぶ。

2.2.3 粒子分布の非一様性・流動性

粒子法では、粒子の分布が計算領域全体に対して非一様であるような計算を扱うことが多い。また、自由表面を有する流れや物体の破壊など、激しい変形を伴う計算を扱うことも多い。このような計算の場合、時間が進むにつれて負荷の均衡が崩れていくことが予想される。

従って、計算全体にわたって負荷バランスを均等に保ち、効率的な並列計算を行うためには、なんらか動的に負荷バランスを調整する必要があると考えられる。

3. 関連研究

本章では、粒子法や、分子動力学、重力多体問題などの並列化における研究例を紹介する。

3.1 直行軸に沿った分割を行う手法

ここで直交軸とは、空間内の x , y , z 軸を指し、これらの軸に直交した平面で領域を分割する方法である。上手く分割平面を決めることで、負荷を均等に分散することができるが、2.2.3 章で述べたように、タイムステップが進むにつれて負荷バランスが崩れていってしまう。

Rogers らによる parallelSPHysics⁷⁾ では、任意の一軸について分割を行い、固定タイムステップ毎に隣接する領域との粒子数の差が閾値以上なら、境界を決められた距離動かすことで、動的な負荷分散を実現している。しかし、この方法では、計算するモデルによって、粒子数の閾値や、境界を動かす距離を選ぶ必要があり、最適値を決めることは難しい。また一軸による分割は、並列度が高くなるにつれて領域が細長く、薄くなるので、並列効率が低くなってしまう。

Fleissner ら¹⁾ は、ORB(Orthogonal Recursive Bisection)⁸⁾ を用いた領域分割を用いている。ORB とは、領域を再帰的に2分割していき、木構造的な分割を行う方法である。領域を2分割する際に、2つの領域の負荷が均等になるようにうまく境界をきめることで、負荷をバランスさせることに成功している。この手法では、固定タイムステップ毎に境界を動かすことで動的に領域を変化させる。このとき、境界の位置を決めるために、粒子をサンプリングしてその分布を推測することで、従来の木構造的手法に比べて領域分割を軽量に行っている。しかし、これらの再帰的アプローチは、根に近いほど操作が重く、また一般に並列度が低くなるので、高並列環境では性能が出にくいことが予想される。

3.2 インデックスソートを用いる方法

粒子法におけるインデックスソートでは、まず計算領域を覆うような格子を考えて、各粒子を、所属する格子の番号によってソートする。次に各格子に、先のソートされた配列を使って、所属する粒子へのリファレンスを保存しておく。この操作によって、局所性を持った粒子配列をつくる。この粒子配列をプロセッサ個に均等に分割することで、局所性を持ったまま均等に負荷をバランスさせることができる。Ihmsen らは³⁾、格子のデータ構造に、スペースフィリングカーブを用いることで、局所性を高めている。

3.3 重心ボロノイ分割を用いた手法

ボロノイ分割は、距離空間に任意の位置に配置された複数の点によって領域分けを行う手法である。計算幾何学における近傍点探索問題等でよく用いられる他、有限要素法の自動メッシュ生成などにも応用されている。

重心ボロノイ分割は、ボロノイ分割を応用した手法であり、科学技術計算においても、N

体問題や分子動力学への適用例がみられる。^{4),9)} ボロノイ分割では、空間内の任意の位置に配置された複数の点 (参照点) に対して、他の点がどの参照点に近いかによって領域分けを行う。つまり、領域は参照点のみによって規定され、各粒子はもっとも近い参照点を持つ領域に属する。すなわち参照点の集合 $P = \{p_1, p_2, \dots, p_n\}$ が存在するとき、参照点によって規定される領域 $V(p_i)$ は、距離関数を d として以下のように表せる。

$$V(p_i) = \{p | d(p, p_i) \leq d(p, p_j), j \neq i\} \quad (2)$$

ボロノイ分割によって、規定された領域の境界線/面は2つの参照点を結んだ直線の、垂直二等分線/面となる。

重心ボロノイ分割では、ボロノイ分割における参照点が、担当する領域の重心と一致するように選ぶ。そのような参照点は、参照点を重心に移動させボロノイ分割を行う、という操作を反復することで得ることができる。この操作はクラスタリングにおいて頻繁に用いられる k-means 法 と同様の方法である。

重心ボロノイ分割の特徴として以下の2点が考えられる。

- 分割が座標軸に依存しない
- 粒子の移動に伴って領域を移動させることができる

分割が軸に依存するということは、境界が計算領域の座標軸と並行な平面になるということの意味とする。3.1章で述べたような、何らか直方体領域を生成する方法では軸に依存した分割を行っている。

粒子法は、複雑な形状の固体や流体を扱い、しかもそれらが大変形を伴う場合の多いため、計算領域の座標軸に依存した領域分割に比べて、座標軸に依存せず計算対象領域のみに依存して領域を定める重心ボロノイ分割はより自然な分割であると考えられる。

また、領域を粒子の重心によって決定しているために、領域内の粒子の移動に伴って参照点が移動し、領域内の粒子の大きな動きに追従するように領域が移動する。従って、静的な領域分割に比べて領域間の粒子の移動が少ないと考えられる。また、粒子の存在する領域において密度が一樣であるような場合、同じ理由によって、ある程度まで均等な負荷分散を保つことができる。

しかしながら、当然負荷バランスは十分に均等化されるわけではなく、また領域移動は領域内の粒子の重心で近似されているために、完全に追従できるわけではない。

そこで、負荷バランスを均等に是正するためのアプローチとして、ボロノイ分割を拡張して重みのパラメータを導入し、粒子と参照点の距離による関係式でなく、距離に重みを加えた4次元の関係式を用いて領域の拡大・縮小を試みる手法が研究されている。以下に、代表

的な2手法を挙げる。

3.3.1 Koradi らの手法⁴⁾

Koradi らの手法では、隣接領域との負荷の差異をもとに重みの値を動的に変更しながら計算を進めていく。ここでの重みは、隣接領域に比べて負荷が大きい場合には参照点と粒子の距離に適当な値を足して全体として領域の境界を狭め、負荷が少ない場合には逆の操作を行って境界を広げるように用いる。具体的には、各領域 V_i に対して重み w_i を考えて、以下のような式 (3) に従って分割を行う。

$$V_i = \{x | x \in V, \|x - x_i\|^2 - w_i \leq \|x - x_j\|^2 - w_j, \forall j \in [1, n]\} \quad (3)$$

重み w_i が大きくなると、領域が大きくなる。このとき w_i は、 a_i を近傍領域の粒子数の平均として、以下の式 (4) に従って、決める。

$$w'_i = w_i + \alpha r_i^2 \left(\left(\frac{a_i}{n_i} \right)^{\frac{2}{3}} - 1 \right) \quad (4)$$

このように重みを簡単な数値計算によって決めるが、係数の α には任意性があり、論文の中では問題に対して、ヒューリスティックに決定されている。この重みの選び方は負荷分散に対して非常に繊細であり、負荷バランスの均衡を左右する。従って、大変形を扱い、粒子が激しく動く粒子法ではこのような数値計算による重みの決定は困難である。

3.3.2 Yahagi らの手法⁹⁾

各プロセッサに以下の式 (5) のように与え、重みを大きくすることで領域を縮小する。

$$V_i = \{x | x \in V, \|x - x_i\|^2 + w_i^2 \leq \|x - x_j\|^2 + w_j^2, \forall j \in [1, n]\} \quad (5)$$

重み w_i の値を大きくすると領域を相対的に小さくすることができる。この手法では、各イテレーションの計算は最も負荷の高いプロセッサに律速されることを鑑みて、各イテレーションにおいて周囲に比べて負荷の高いプロセッサ (Locally Load Maximum, LLM) が負荷を大きくして、隣接領域に負荷を分け与えることを目標にしている。

具体的には、以下のアルゴリズムによって計算される。

If the processor is the LLM,

calculate with the signed distances of the particles to the nearest domain border.

sort their signed distances in a descending order.

read the sorted distance in the $[(Load/Load_{here})n_{particle}]$ -th row.

redefine w_{here} as $w_{here} +$ above distance.

End of if

この重みの決定は、LLM であるような領域が、平均粒子数を持つように、境界を更新することを主眼としている。しかし、重みの更新による境界の変位は、更新前の自プロセスの重みの他に、各隣接領域の重み、各隣接領域の参照点との距離に依存するために、厳密に平均粒子数となるように重みを決定することは困難であるため、近似的に行われている。

また、このように LLM のみが重みを変更する場合、LLM である領域の隣接領域が、同じ程度の粒子数を持っているときには上手く粒子が分散されることが予想されるが、隣接領域間での粒子数が異なる場合、特に LLM と粒子数がほぼ変わらないような領域が隣接していた場合には、かえって最大粒子数を増やしてしまう可能性がある。つまり、高負荷領域が隣接していた場合には、一種の振動状態に陥ってしまい、負荷バランスをかえって悪化させてしまう可能性があるということである。特に、毎タイムステップ粒子が激しく動く粒子法では、この種の可能性が大きくなると考えられる。

4. 提案手法

本章では、提案手法について述べる。

4.1 基本アイデア

提案手法では、3.3 章でみた重心ボロノイによる領域分割をもとに、その負荷バランスの不均衡を是正することを目標とする。

3.3.1, 3.3.2 章でみた、重みによる負荷分散の是正方法の問題点は、領域の境界を平均的に拡大、もしくは縮小していたことであつた。即ち、負荷の大きい領域の粒子数を平均粒子数に近づけることのみを目標にしていたため、高負荷領域が隣接しているような場合に、かえって負荷バランスを悪化させてしまうことが予想された。従って、提案手法では、隣接領域の負荷に応じて分配する粒子数を調整することで、全体として平均化を図ることを目標とした。

具体的には、以下のように行う。

各領域は自領域の粒子数を、自領域と隣接領域の平均粒子数と比較し、自領域の粒子数が平均より過剰であれば、隣接領域に粒子を分配する。このとき、自領域の粒子数と平均の差分の数の粒子を、隣接領域のうち平均よりも粒子数が少ない領域に対して、平均よりもどれだけ少ないかの比に応じて分配する。

図 1 は、ボロノイ分割によって分割された領域に、提案手法の適用した例を示した物である。図中では、4 つの状態を描いており、左側の色のついた領域に着目している。以下に、それぞれの状態について簡単に説明する。

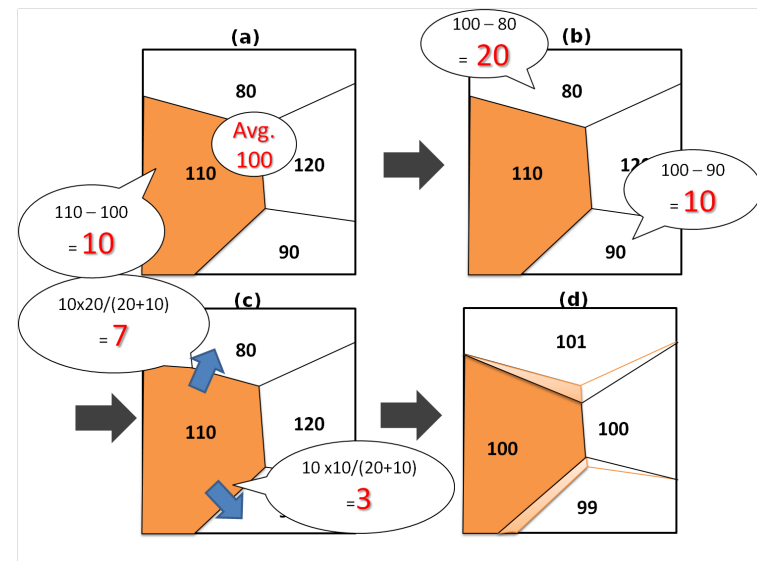


図 1 提案手法による負荷分散のイメージ。4 つの領域に分かれており、図中では左側の領域によって行われる計算を示している。右下の 4 番目の図は、各領域によって負荷分散がなされた後の状態を示している。

- 自領域とその隣接領域の平均粒子数は 100 である。自領域の粒子数は 110 なので平均 100 より多く、その差分である 10 を分配する。
- 分配すべき領域は平均である 100 より少ない上下の領域であり、それぞれ平均の 100 との差は、20, 10 である。
- (b) で求めた数から、粒子の分配比は 2:1 である。従って、この比に従って (a) で求めた 10 を分けると、上下にそれぞれ 7 と 3 である。
- 上の (a),(b),(c) の操作を各領域について行った結果を示している。

粒子のやりとりを隣接領域としか行わないのは、隣接していない場合、自分の担当する粒子と領域との間に別の領域が入ってしまい、飛び地ができてしまう可能性があるためである。また、粒子の受け渡しは一方向に限定される必要がある。これも、領域の飛び地ができてしまう可能性があるからである。今回は粒子の受け渡しは粒子数の大きい領域から小さい領域のみとした。

このアルゴリズムは、各領域は局所的な平均を目指し、その結果として全体として負荷の

平滑化がなされるというものである。従って、複数回適用することでより広い領域での平均化がなされると考えられ、全体の平均化も進むことが期待される。

次章にアルゴリズムの詳細を示す。

4.2 アルゴリズム

計算領域全体に粒子が N 個存在し、プロセッサの数が p 、領域全体が p 個の小領域 $D_1, D_2, \dots, D_p$ に分割されているとする。また、領域 D_i は n_i 個の粒子を含んでいるとする。すなわち $\sum_{i=1}^p n_i = N$ である。また、領域 D_i と境界を共有する隣接領域の数を f_i とし、 D_i の隣接領域を $D_{a_1}, D_{a_2}, \dots, D_{a_{f_i}}$ とする。

このとき、全ての D_i について以下のプロセスを行う。

- (1) n_i 及び $n_{a_1}, \dots, n_{a_{f_i}}$ の平均を m_i とする。これは、自領域と隣接領域の粒子数の平均である。即ち $m_i = \frac{n_i + \sum_{j=1}^{f_i} n_{a_j}}{f_i + 1}$ である。
- (2) n_i と m_i の差を求め、 e_i とする。これは自領域の粒子数と、平均粒子数の差を求めており、 $e_i = n_i - m_i$ である。
- (3) 差 e_i が負なら終了する。これは、自領域の粒子数が隣接領域の平均より多いときには隣接領域に粒子を譲り渡して平均化を図り、平均より少ないときには隣接領域から譲り受けて平均化を図るためである。
- (4) $m_i > n_{a_j}$ であるような D_{a_j} を f'_i 個とし、これを満たすような D_{a_j} を $D'_{a_1}, D'_{a_2}, \dots, D'_{a_{f'_i}}$ とする。
- (5) 各 D'_{a_j} に対して、 D_i が渡すべき粒子の数 e_{a_j} を求める。隣接領域の平均粒子数よりも少ない粒子数を持つような領域に対して、平均よりもどれだけ少ないかの比に応じて、粒子数を分配する。従って、求める数 e_{a_j} は次のようになる。
$$e_{a_j} = \frac{m_i - n_{a_j}}{\sum_{k=1}^{f'_i} m_i - n_{a_k}}$$
- (6) 各 D'_{a_j} に対して、 D_i から e_{a_j} 個の粒子を送る。 e_{a_j} 個の選び方は、 D_i に含まれる粒子の中で、 D_i を除いて D'_{a_j} が一番近いようなもののうち、 D'_{a_j} に近いものから e_{a_j} 個とする。もしそのような粒子が e_{a_j} 個に満たないようであれば、存在する分のみを送る。

5. 実装

本章では、提案手法の実装に関する概要と、要素技術を述べる。実装には C++ と、MPI(Message Passing Interface) の代表的な実装である MPICH2 を用いた。

5.1 空間格子の利用

提案手法では上で述べたように粒子の受け渡しのために、各領域において各隣接領域に一番近い様な粒子を管理しておく必要がある。さらに、ある隣接領域が一番近いような粒子集合は受け渡しの際により近いものから選ばれるので、これらの粒子は距離によってソートされている必要がある。この管理のためのメモリコスト及びソートの計算負荷を軽減するために、領域分割及び負荷分散における粒子管理の単位として空間格子を導入した。

空間格子は、粒子法や分子動力学における近傍粒子探索で頻繁に用いられる方法であり、計算領域を立方体格子に区切って、粒子を登録して管理する方法である。この空間格子の内部領域を今後セルと呼ぶことにする。

5.2 隣接領域の管理

粒子法では、外点や移動にともなって担当の領域から外れた粒子などの情報を送受信するため、隣接領域の情報を管理する必要がある。すなわち、あるセルがどの領域に所属しているのかを算出できなければならないということである。ボロノイ分割では、あるセルが属する領域は参照点を用いて算出することができるが、提案手法ではボロノイ分割による数学的な規則性を保持しないため、負荷分散によって分配されたセルは参照点のみでは算出できず、なんらか別の方法で管理する必要がある。以下に、今回用いた管理手法を述べる。

5.2.1 近傍領域の決定

提案手法では、外点やみ出し粒子、負荷分散のための粒子の譲渡の為に通信を行う可能性のある領域を、近傍領域として各タイムステップの初めにあらかじめ算出しておくことで、計算量を削減している。近傍領域の判断には、領域の最大半径を使用した。即ち、各領域は自分の担当する粒子と参照点の距離の中で、最も長いものを領域半径とする。このとき、各領域内に存在する粒子は、参照点を中心として、領域半径を半径とした円の中に含まれる。従って、任意の2つの領域について、この円が重なっていれば、領域は近傍であると判断する。図2では、図の中心に存在する領域の近傍領域を色付きで示している。

ただし、実際には、円が重なっていなくても、2領域の最も近い境界がカットオフ半径以下であるようなときに、通信が行われる可能性があることに注意する。

近傍粒子の判断は、領域 D_a と D_b の参照点を g_a, g_b 、半径を r_a, r_b 、カットオフ半径を r_e とすると、下の式(6)を満たすとき、 D_a と D_b は近傍であると判断される。

$$|g_a - g_b| < r_a + r_b + r_e \quad (6)$$

このため半径情報は、参照点情報と共に毎タイムステップ、全プロセスで共有されている。

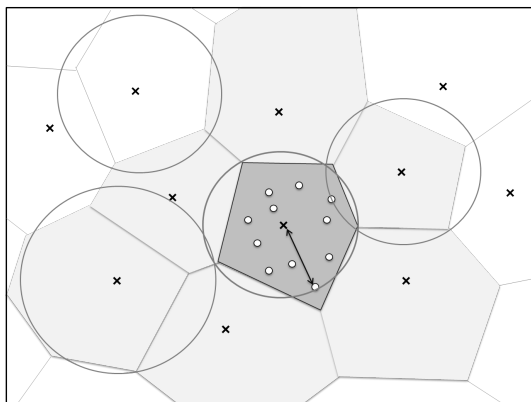


図 2 近傍領域決定のイメージ。図中心の領域の近傍領域が色付きで示されている。図中の × は参照点、円は領域の参照点を中心とし、領域半径を半径とした円。この円が重なった場合に近傍と判断する。

従って、上の式 (6) の結果は全プロセスで同じであることが保証されている。なお、この方法では実際には通信を行う可能性の無い領域を近傍と判断する場合があるが、計算の正確性や性能に影響を与える性質のものではない。

5.2.2 セルの所属情報の管理

5.2 章で述べたように、粒子法では各領域が、外点の所属領域を算出できる必要がある。ボロノイ分割のみによる領域分割では各領域の参照点情報を用いて、任意の座標がどの領域に属するかを算出することができるが、提案手法による負荷分散ではボロノイ分割後に別の規則によってセルを分配するため、ボロノイ分割による法則性が保たれない。つまり、提案手法による負荷の分配によって所属が変わったセルは、ボロノイ分割の関係式を用いて所属領域を算出することができないということである。

従って、提案手法による負荷の分配によって所属が変わったセルについては、そのセル ID と所属領域の対応情報を、近傍領域間で共有することによって解決する。この対応情報は、負荷分散の際に粒子の分配を行った領域が近傍領域に伝達することで共有される。この情報によって、各領域は近傍領域におけるセルの所属変更を確実に知ることができる。図 3 では、領域番号 0 (rank0) がセルの分配を行い、その情報を近傍領域に伝達する様子を表している。

従って、提案手法で各領域が自分の担当外のセルの所属を計算するときには、まずセルと

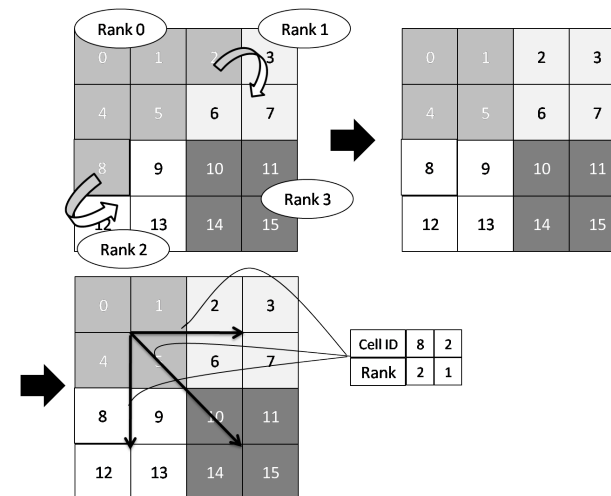


図 3 セルの所属情報管理のイメージ図。図の rank 0 が提案手法によるセルの分配を行い、その後、行われた分配の情報を近傍領域に伝達している。以後、各領域はセルの所属領域を算出するときには、まず近傍領域から送られたこの情報を参照することになる。

領域の対応表を探索し、対応表にそのセル ID が無ければ参照点を用いて算出することになる。このとき、所属に変更のあったセルの数は各領域が担当するセル数に比べて十分少ないため、この対応表を検索する部分の計算が全体の効率に影響することはほとんど無いと予想される。

6. 性能評価と考察

本章では、提案手法による実験の評価と性能に関する考察を行う。評価を行う対象は負荷分散、逐次計算を基準としたスケーラビリティとし、3.3.2 で述べた手法を比較対象とした。

6.1 SPH

今回提案手法を評価するに当たって、SPH(Smoothed Particle Hydrodynamics Method) を使用し、非圧縮性流体の計算を行った。SPH の実装に関しては、カーネル関数は Muller ら⁵⁾ によるもの、圧力の導出には理想気体の状態方程式、境界(壁)の作用には単純な弾性衝突を用いた。

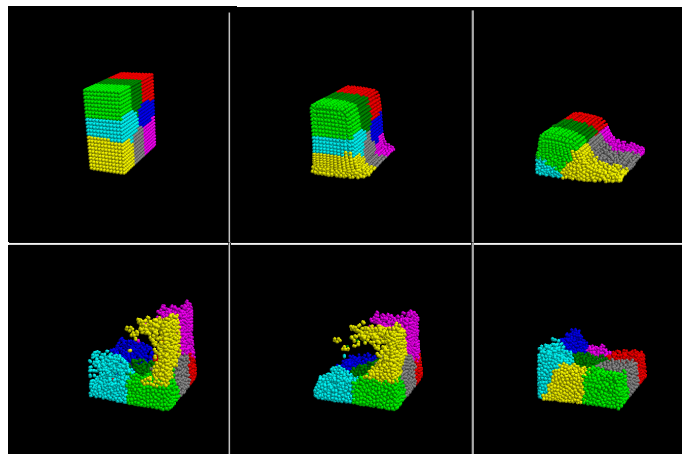


図 4 約 1 万粒子によるダム崩壊問題での提案手法による負荷分散の様子。粒子の色によって所属領域を表している。

表 1 実験環境

環境名	ノード数	CPU	メモリ	ネットワーク	OS
huscs	16	Xeon E5530(2.40GHz)×8	24GB	10Gbit Ethernet	2.6.26-2-amd64

6.2 実験条件

実験する計算モデルは、粒子法による流体計算の評価において一般的なダム崩壊問題である。図 4 はダム崩壊問題における提案手法の適用結果を、可視化して示したものである。実験環境は表 1 の通りである。

なお、実行時間を計測する際には、各タイムステップでの粒子座標の出力は行っていない。

6.3 負荷分散

提案手法と既存手法の負荷分散の様子をグラフに示した。図 5 は、41 万 4239 粒子によるダム崩壊問題を 8 プロセス及び 128 プロセスで実行した際の各プロセッサの粒子数をプロットしたグラフである。

比較対象の手法に比べて、提案手法では粒子数の分散が小さく収まっている様子が観察できる。

6.4 スケーラビリティ

提案手法と既存手法に関して、逐次プログラムを基準としたスケーラビリティを図 6 に示した。128 プロセスでのスピードアップは両手法ともに 52 倍前後であった。全ての並列

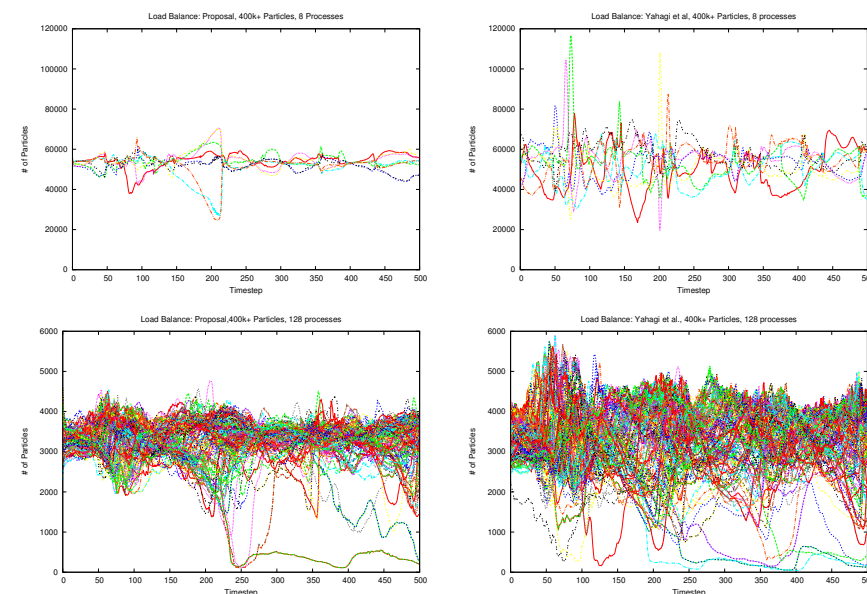


図 5 約 41 万粒子によるダム崩壊での負荷分散の様子。縦軸が粒子数、横軸が時間である。左側が提案手法、右側が比較手法。上段が 8 並列、下段が 128 並列である

度において提案手法が上回ったが図からわかるとおり、32、64 並列以外では明らかな差は無かった。

6.5 考察

提案手法では、できるだけ均等な負荷バランスを低オーバーヘッドに維持することで、毎タイムステップの実行時間の短縮を目指している。まず各タイムステップでの負荷バランスを見てみると、提案手法は Yahagi らの手法と比較して、計算全体を通して粒子数の分散が小さく、優れた負荷分散を行っていると考えられる。

ところで、提案手法と Yahagi らの手法における負荷分散のための計算によるオーバーヘッドは、ほぼ同じはずである。従って両者で負荷分散に差があるならば、より均等な負荷分散を行っている手法の方が高速であるはずである。スケーラビリティを計測した際の負荷分散は、2、4 並列では殆ど同じ程度であり、8 以上の並列度では提案手法の方がより均等な負荷バランスとなっていた。しかし図 6 を見てみると、32、64 並列の時には明らかな差がある

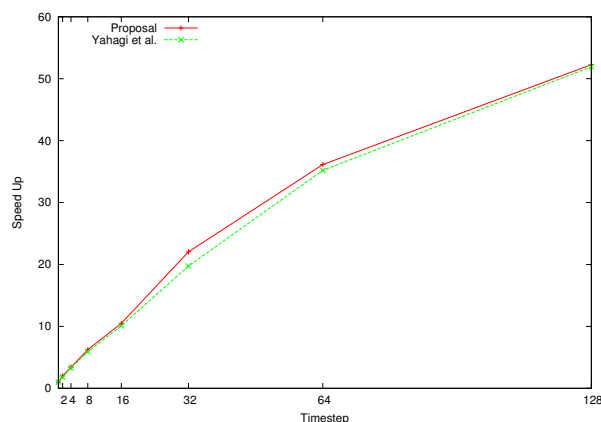


図 6 約 41 万粒子によるダム崩壊問題での、1-128 プロセスのスケールビリティの比較

ものの、他の場合には殆ど差が出ていない。

この理由として、並列度が大きいときに関しては、1 プロセッサあたりの問題の規模が小さくなったために、相対的に粒子数の違いが隠蔽されたのだと考えられる。従って、大きな並列度においても十分に多い粒子数が存在する場合には顕著な差が現れると考えられる。

並列度が小さい時に関しては自明ではなく、今後さらなる調査が必要である。

7. おわりに

7.1 ま と め

本研究では、並列粒子法において重心ボロノイ分割を用いた、動的に負荷バランスを是正することのできる手法を提案し、実験と評価を行った。その結果、十分に均等な負荷分散を得ることに成功し、分散環境において 128 プロセスで約 52 倍の速度向上を得ることができた。

7.2 今後の課題

今後に向けて、以下のような課題が挙げられる

- 詳細かつ広範な評価

最大 128 並列、粒子数は 40 万程度、ダム崩壊のみ、と限られた評価しか行えていない。パラメータを変えて、様々な規模、モデルによる計算を行い、提案手法の性能を明らかにしたい。

- 他の手法との比較

今回比較した手法は、現在並列粒子法において広く使われている方法ではない。今後は、実際に用いられている手法を実装し、比較・評価する必要がある。

- 実装の最適化・高速化

今回の実装には、高速化の余地が多く残されている。キャッシュを意識したデータ構造や、計算環境のアーキテクチャを考慮した領域の割り当て等のチューニングを行うことで、より良い成果が期待されると考えている。

参 考 文 献

- 1) Fleissner, F. and Eberhard, P.: Load Balanced Parallel Simulation of Particle-Fluid DEM-SPH Systems with Moving Boundaries., *PARCO'07*, pp.37–44 (2007).
- 2) Harada, T., Koshizuka, S. and Kawaguchi, Y.: Smoothed Particle Hydrodynamics on GPUs, *Structure*, pp.1–8 (online), available from <http://individuals.iit.u-tokyo.ac.jp/~yoichiro/report/report-pdf/harada/international/2007cgi.pdf> (2007).
- 3) Ihmsen, M., Akinci, N., Becker, M. and Teschner, M.: A Parallel SPH Implementation on Multi-Core CPUs, *Computer Graphics Forum*, Vol.30, No.1, pp.99–112 (2011).
- 4) Koradi, R., Billeter, M. and Gntert, P.: Point-centered domain decomposition for parallel molecular dynamics simulation, *Computer Physics Communications*, Vol.124, No.2-3, pp.139–147 (2000).
- 5) Müller, M., Charypar, D. and Gross, M.: Particle-based fluid simulation for interactive applications, *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '03, Aire-la-Ville, Switzerland, Switzerland, Eurographics Association, pp.154–159 (2003).
- 6) Oger, G., LeTouz, D., Alessandrini, B. and Maruzewski, P.: A new parallelized 3D SPH model: resolution of water entry problems and scalability study, *ERCOFTAC Bulletin*, Vol.76, pp.35–38 (2008).
- 7) Rogers, B., Dalrymple, R., Gmez-Gesteira, M. and Crespo, A.: *User Guide for the parallelSPHysics Code using MPI v2.0* (2011).
- 8) Salmon, J.K.: Parallel hierarchical N-body methods, PhD Thesis, California Institute of Technology (1991). UMI Order No. GAX91-37285.
- 9) Yahagi, H., Mori, M. and Yoshii, Y.: The Forest Method as a New Parallel Tree Method with the Sectional Voronoi Tessellation, *The Astrophysical Journal*, Vol.124, p.1 (1999).