

## 分子動力学シミュレーションの並列化における分割法の一考察

西村 健† 中村 周吾‡ 池口 満徳‡ 清水 謙多郎‡

† 東京大学大学院 理学系研究科 情報科学専攻

‡ 東京大学大学院 農学生命科学研究科 応用生命工学専攻

### 要旨

従来、分子動力学シミュレーションの並列アルゴリズムとして、負荷分散を目的とする原子分割法と通信量の最小化を目的とする空間分割法が提案されてきた。本研究では、負荷分散と通信量の低減の双方を考慮した、新しい並列アルゴリズムをいくつか提案する。これらのアルゴリズムを水中のタンパク質分子 (16735 原子) のシミュレーションに適用し、性能を比較したところ、中でも、微小セルを単位として原子を並べ、それらに働く力の計算を均等に分割してプロセッサに割り当てるアルゴリズムが最も良い性能を示し、従来の空間分割法に比べて 10% 以上の性能向上が達成されるという結果を得た。

## Decomposition Methods for Parallel Molecular Dynamics Simulation

Takeshi Nishimura† Shugo Nakamura‡ Mitsunori Ikeguchi‡ Kentaro Shimizu‡

† Department of Information Science, Graduate School of Science, University of Tokyo

‡ Department of Biotechnology, Graduate School of Agricultural and Life Sciences, University of Tokyo

### Abstract

Many parallel algorithms for molecular dynamics simulation have been proposed. Most of these algorithms can be classified into atom decomposition which aims at load balancing and space decomposition which aims at the minimization of communication cost. This paper proposes several new hybrid algorithms which consider the both. We evaluate the performance of these algorithms as well as conventional atom decomposition and space decomposition for a protein molecule in water (which consists of 16735 atoms in total). Among these hybrid algorithms, small-cell based force decomposition algorithm shows the best performance; it achieves more than 10% speedup compared with the conventional space decomposition algorithm.

## 1 はじめに

分子動力学法とは、多原子系における原子の運動を、個々の原子に対し、まわりの原子との相互作用（力）を計算しながら、Newtonの運動方程式を積分して求めるというものである。このような積分は解析的に計算することが不可能であるため、有限差分法を用いて数値的に計算する。分子動力学法の基本手順は有限差分法の各タイムステップにおいて力の計算、原子座標の更新を繰り返し、時々刻々の原子の位置・速度を計算するというものである。

分子動力学法は、例えば、分子生物学の分野においては、生体分子の構造やダイナミクスの理解、生体分子の機能予測や解析など、さまざまな分野で利用されている [1]。このような利用において、とくに生体内に近い環境で精度の高いシミュレーションを行うには、水中のタンパク質のような、大規模で、原子分布が不均一な系での計算が不可欠である。

このような計算を精密に行おうとすると膨大な時間を必要とする。そのため、近年、分子動力学法の並列アルゴリズムの研究が盛んに行われるようになった。分子動力学法の1ステップの計算は原子ごとに独立に行なうことができるため、単純な並列化は容易である。ただし次のステップに移るときに、次のステップでの計算に必要なデータを通信する必要がある。並列化をいかに効率良く行なえるかは、どの原子の計算をどのプロセッサで行なうかという原子群の分割法に依存するところが大きい。

従来の分割法には、大きく分けて、原子分割法と空間分割法の2つがある。また、計算に必要なデータ（力および原子座標）の通信には、ブロードキャスト通信を行うものと個々のプロセッサ間で通信するものがある。

原子分割法 [2] は、各プロセッサが担当する原子の数が一定になるように、原子をプロセッサに割り当てる方法である。各プロセッサの計算量の均等化（負荷分散）を目指す。原子分割法では、どの原子をどのプロセッサに割り当てるかということとはとくに規定していないため、1

プロセッサが分子動力学シミュレーションの計算のために必要とするデータは他のプロセッサに分散する傾向にあり、通信にはブロードキャストが用いられるのが一般的である。逆に全てのデータをプロセッサ間で共有するため、原子の割り当てを任意にすることができるとも言える。

これに対して空間分割法 [3] は、なるべく通信量を減らすという方針を採り、空間を等間隔の格子で分割し格子で仕切られた領域をプロセッサに割り当てる。この方法では、シミュレーションの計算に必要なデータの通信は限られたプロセッサとの間で行なえば良くなるため、ブロードキャストを用いるより通信量を削減することができる。

我々は、通信量の最小化と計算量の分散の両方が分割法の重要な指標であると考える。計算量が分散していなくてあるプロセッサにおける計算量のみが大きいと、次ステップの計算に移ることができず、全体の計算時間が増大する。一方通信量の最小化はそのまま通信時間の最小化につながる。原子分割法は任意のプロセッサで任意の原子の計算が行なえるため計算量の分散は行なえるが、ブロードキャストを用いるため通信量の大きさが問題となる。空間分割法は原子分割法と比較して通信量の削減は行なえるが、原子分布が不均一な系では計算量の分散にばらつきが生じるという問題がある。

本研究では、通信量を削減しつつ各プロセッサでの計算量が等しくなるような方法を実装し、比較のため実際の並列計算機上で実験を行なった。タンパク質と水の系で実験を行なったところ、従来の方法と比較して1割程度の速度向上を達成した。

## 2 分子動力学シミュレーションにおける分割法

分子動力学シミュレーションにおいて並列計算を効率良く行なうためには、通信量の削減と計算量の分散が重要である。通信量を削減するためには、力の計算に必要なデータが同じブ

ロセッサ上にあることが非常に重要である。つまり、近傍にある原子は、できるだけ同じプロセッサに割り当てなければならない。一方、各プロセッサが担当する計算量を均等化させるには、プロセッサに割り当てる原子の数を等しくする必要がある。

しかしながら、通信量を最小化することと計算量を均等化することの両方を同時に達成することは一般には不可能である。つまり、通信量を削減するために原子を移動することは移動先のプロセッサの計算量を増加させることになり、逆に計算量を等しくする操作は通信量を増加させることになる。

そこで本研究では、通信量の低減と計算量の分散の両方を適度に考慮したさまざまな分割法を設計・実装し、実際の並列計算機上で性能評価を行うことによりその有効性を調べる。

次節で説明する我々の方法は、3つの方法に大別される。

まず1つ目の方法は、計算量を分散させるために、原子を一次元に並べ、それをプロセッサ台数分に等しく分割し、各プロセッサへ割り当てる。一次元に並べるときに、空間的に近いもの近くなるように並べれば、ある程度の通信量の削減も同時に達成される。

2つ目の方法は、空間を平面により分割するが、その時に2つの部分空間に等しい数の原子が存在するようにする。それぞれの部分空間を同様に分割し、分割数がプロセッサ台数に等しくなるまで繰り返す。このようにすれば、通信が必要な原子は切断平面の近傍の原子に限られるため、通信量は抑えられると考えられる。

3つ目の方法は、通信量を削減するために、平均距離が最小となるようなクラスタ分割を行なう。加えて、クラスタに含まれる原子数を一定以上にならないように抑えることによって、計算量の分散を図る。

### 3 通信量と計算量を考慮した分割法

以下、我々が比較を行なった6種類の方法について説明する。なお、以下で  $N$  は原子数、 $P$

はプロセッサ台数を表すものとする。

#### 1. 原子一次元分割 (FILE)

ファイルに格納されている原子順をそのまま用いて1次元に並べ、それを原子数が等しくなるように端から順に  $N/P$  個ずつ1つのプロセッサに割り当てる。最も単純な方法であるといえる。ファイルには、残基などの情報がまとめて入っていることが多いため、この原子順を用いた場合でもある程度原子分布を考慮した分割が行なわれることが期待される。すなわち、力の計算に必要なデータが同じプロセッサに集まり通信量が削減されるということである。従来の原子分割法とはブロードキャストをしないという点で異なる。

#### 2. セル一次元分割 (CELL)

原子一次元分割と同じように、あらかじめ空間を空間分割法と同様にセルと呼ぶ微小領域で分割し、そのセルを一次的に並べて、原子の並びを決定する。各セルは、まず  $X$  軸方向につなげてそれを端のセルで  $Y$  軸方向につなげる。これで  $XY$  平面上のセルは1つに並べることができる。さらにその端のセルを  $Z$  軸方向につなげて1つの並びとする。セル内での原子の並びについてはとくに定めない。原子一次元分割に比べてより近傍の原子が集まることを期待される。ただし、原子一次元分割と同じく一次的に並べることによる原子の分散は回避されない。

#### 3. 最小距離分割 (MIND)

原子を1つ取り出し、他の原子の中からその原子との距離が最小であるものを1つ選ぶ。次に、今度は選ばれた原子を起点として同じことを繰り返す。これを原子の数だけ並べて全原子を一次的に並べ、原子一次元分割と同じく各原子をプロセッサに割り当てる。

#### 4. 部分最小距離分割 (MINDP)

あらかじめ空間を平行移動させた複数枚の

平面で等分しておき、その部分空間の中の原子を最小距離分割と同じ方法で一次元に並べる。それを全てつなげて全体として1つの並びとし、それを原子一次元分割と同様端から順にプロセッサに割り当てる。平面で分断され平面を越えた原子間の距離が考慮されないため最小距離分割よりも一次元に並べた際の隣接原子間距離は大きくなるが、大域的に見るとセル一次元分割のよ

#### 5. 2分割反復法 (CUBE)

2分割反復法は、空間を原子数が等しくなるようにXY平面で2分割し、それぞれの部分空間について分割する平面を変えながら同じことを繰り返し総分割数がプロセッサ数に等しくなるようにするというものである。これにより、各プロセッサに割り当てられる原子数はほぼ等しくなり、空間分割と同程度の通信量が期待される。この方法は、Nylandらによって提案された[4, 5]。

#### 6. 距離クラスタ分割 (CLD), 力数クラスタ分割 (CLF)

クラスタ分割を利用する。距離クラスタ分割の場合は、全ての原子が別々のクラスタからなる状態から始め、まず平均距離が最も短い2クラスタを統合する。これをクラスタ数がプロセッサ数に等しくなるまで繰り返す。これにより、近い原子同士がクラスタを作り、多くの力の計算が1プロセッサ内のみで可能になることが期待される。

力数クラスタ分割は、クラスタ間の平均距離の代わりにクラスタをまたがる力の数を用いてクラスタ分割を行なう。

この変種として、原子数が $N/P$ を越えるクラスタについては以降統合しないという制約をつけた方法 (CLDNO, CLFNO) も考えられ、性能評価の対象とした。

力の計算には数原子のデータが必要であるが、どの原子が割り当てられたプロセッサで力

の計算を実行するかは、力の種類ごとにあらかじめ決めておく。

また、原子一次元分割とセル一次元分割に関しては、原子数を一定にするのではなく計算する力の数を一定にする分割法 (FILEE, CELLE) が考えられる。この方法では、原子をプロセッサに割り当てる時にそれが関係する力もできる限りそのプロセッサに割り当てる。

## 4 実験および考察

我々は、上記の方式をMPIを用いて実装を行った。そして、東京大学大型計算機センターのSR2201において、タンパク質BPTI(Bovine Pancreatic Trypsin Inhibitor)と水の系で実験を行なった。BPTI自身の原子数は892で、この系には水分子を含めて16735原子が存在する。

表1に1ステップの実行にかかる時間とその内訳および空間分割法の実行時間に対する割合を示した。SR2201で64台のプロセッサを用いて計算している。ATOMは原子分割法、SPACEは空間分割法を表し、あとの方法は上記で説明した方法である。力の計算にかかる時間 (calc. time) は各プロセッサでの時間の最大値を採用し、通信時間 (network time) は各プロセッサでの時間の最小値を採用することによって、純粋に計算および通信に要する時間を求めた。原子分割法はブロードキャストを使っており他の方法と処理が異なるためtotal timeのみを示した。speedupは、空間分割法を1とした相対的な速度を示している。

1ステップの実行時間を見ると、力の数を一定にする分割法 (FILEE, CELLE) の結果が総じて良いことが分かる。これは、計算量が各プロセッサに分散され、計算時間の最大値が低く抑えられていることによるところが大きい。つまり、SR2201のように通信が速い並列計算機で、64プロセッサ程度の並列性の場合通信量を削減する分割法よりも計算量を分散させる分割法の方が有効に働くといえる。

距離クラスタ分割は通信量に関しては最小を

| methods         | ATOM   | SPACE  | FILE   | CELL   | MIND   | MINDP  | CUBE   |
|-----------------|--------|--------|--------|--------|--------|--------|--------|
| calc. time(s)   | -      | 0.0911 | 0.0793 | 0.0786 | 0.0809 | 0.0811 | 0.0824 |
| network time(s) | -      | 0.0137 | 0.0264 | 0.0176 | 0.0292 | 0.0255 | 0.0153 |
| total time(s)   | 0.180  | 0.105  | 0.106  | 0.0963 | 0.110  | 0.107  | 0.0980 |
| speedup         | 0.58   | 1.00   | 0.99   | 1.09   | 0.95   | 0.98   | 1.07   |
|                 | CLD    | CLF    | CLDNO  | CLFNO  | FILEE  | CELLE  |        |
|                 | 0.140  | 4.188  | 0.136  | 0.0843 | 0.0788 | 0.0737 |        |
|                 | 0.0175 | 0.0453 | 0.0178 | 0.0224 | 0.0265 | 0.0176 |        |
|                 | 0.157  | 4.234  | 0.153  | 0.107  | 0.0977 | 0.0921 |        |
|                 | 0.66   | 0.02   | 0.68   | 0.98   | 1.07   | 1.14   |        |

表 1: それぞれの分割法による 1 ステップの実行時間とその内訳 (SR2201,64 プロセッサ)

与えることが期待されるが、他の分割に比べて必ずしも最小の通信時間を示していない。これは、距離は通信量を表す間接的な指標であり完全に通信量を反映しているわけではないことも関係するであろうが、それ以上にプロセッサの位置、通信における特性が影響を与えているものと思われる。この原因は今後詳細に検討する必要がある。

距離クラスタ分割は、統合が進むにつれて大きなクラスタ同士を統合してしまうためクラスタ間のばらつきが大きくなる。計算量を分散させるという点においては劣っているといえる。

原子次元分割においては、タンパク質を構成する原子を割り当てられたプロセッサの計算時間が大きくなる傾向を見せた。これは、タンパク質部分では原子が密集し原子 1 個あたりの計算量が大きいため、原子数で分割することが必ずしも計算量を一定にするものではないことを示している。

力数クラスタ分割はそれ自身ではあまり良い分割を与えないが、原子数の制限を加えることによってより良い分割を与える。これは、クラスタが統合されるのが概ね原子数の大きいものからで、原子次元分割や他の次元分割と同じように大きいクラスタに小さいクラスタを次々に統合していくという過程を通るからである。

最小距離分割は、初めのうちはある程度空間

的にまとまった原子が 1 プロセッサに割り当てられるが、最後には空間全体に広がるまばらな原子が 1 プロセッサに割り当てられてしまう。これが通信時間の増加という形で表れている。どの原子から分割を始めるかにも依存するが、他の方法と比較して通信量の増加は免れないであろう。

次に、プロセッサ台数の変化による 1 ステップ実行時間の変化を空間分割法からの相対速度として図 1 に示す。ここに挙げているのは、64 プロセッサのときに空間分割法よりも速かった 4 つの方法のみである。

この結果から、これらの 4 つの方法は、プロセッサ台数によらず、従来の方法より良い結果を与えていることが分かる。

参考のため、それぞれの分割の計算に要する時間は表 2 のようになる。ただし現在は分割の部分は並列化されていないため、並列化できればより分割に要する時間は短縮されると考えられる。なお、クラスタ分割の各方法はあまりにも分割に要する時間が長過ぎるため除外した。

## 5 関連研究

分子動力学法の並列アルゴリズムは、これまで多数のものが提案されてきたが、それらは、基本的に原子分割法が空間分割法に属するものである。別に力分割法 (force decomposition) と呼ばれる方法もあるが、これは原子分割法の一

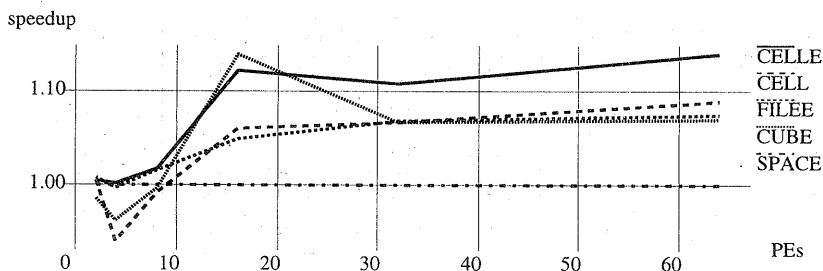


図 1: プロセッサ台数と 1 ステップの実行時間

| methods               | SPACE | FILE  | CELL  | MIND | MINDP | CUBE  |
|-----------------------|-------|-------|-------|------|-------|-------|
| decomposition time(s) | 0.376 | 0.368 | 0.374 | 78.6 | 11.3  | 0.829 |
|                       | FILEE | CELLE |       |      |       |       |
|                       | 3.97  | 4.01  |       |      |       |       |

表 2: 各分割法の分割に要する時間

種で、更に計算量を分散させるためのものと解釈できる。本研究のように、これらの折衷方式を提案したものとしては、Srinivasan らによる 2 分割反復法を用いた負荷分散方式 [5] が挙げられる。彼らは、分子の移動に伴う動的な負荷分散についても検討し、2 分割反復法が容易に対応できることを示しているが、通信量を削減するという点についてはほとんど触れられていない。

Nyland らは、2 分割反復法と原子一次元分割法およびその力均等版について、共有メモリ型並列計算機でのシミュレーションの実行時のコストを理論的に見積もっている [4]。しかしながら、本研究のように分割法を広範に検討したのではなく、また、その見積りは、分散メモリ型計算機にそのまま適用できるものではない。

## 6 まとめ

我々は、分子動力学シミュレーションの並列化において、原子をプロセッサに割り当てる分割法について考察し、従来の原子分割法と空間分割法の折衷方式として、負荷分散と通信コストの最小化の両方を目的とした新たな分割法を設計・実装した。これにより、タンパク質と水

の系において 1 割程度の速度向上を達成した。

今後、原子数を元にするのではなく力の数を元にした分割法について考察を進めるとともに、通信時間の詳細について分析しなければならない。また、分割法自身の並列化の考案、およびシミュレーションの進行に伴う原子の移動への分割の対応を行いたいと考えている。

## 参考文献

- [1] Karplus, M. and Petsuko, G. A.: Molecular dynamics simulations in biology, *NATURE*, Vol. 347, pp. 631-639 (1990).
- [2] Smith, W.: Molecular dynamics on hypercube parallel computers, *Comp. Phys. Comm.*, Vol. 62, pp. 229-248 (1990).
- [3] Plimpton, S.: Fast Parallel Algorithms for Short-Range Molecular Dynamics, *Journal of Computational Physics*, Vol. 117, pp. 1-19 (1995).
- [4] Nyland, L., Prins, J., Yun, R. H., Hermans, J., Kum, H.-C. and Wang, L.: Achieving Scalable Parallel Molecular Dynamics Using Dynamic Spatial Domain Decomposition Techniques, *Journal of Parallel and Distributed Computing*, Vol. 47, pp. 125-138 (1997).
- [5] Srinivasan, S. G., Ashok, I., Jónsson, H., Kalonji, G. and Zahorjan, J.: Dynamic-domain decomposition parallel molecular dynamics, *Comp. Phys. Comm.*, Vol. 102, pp. 44-58 (1997).