TSUBAME 共同利用　平成28年度　学術利用　成果報告書

# 利用課題名　ポストペタ時代の大規模並列数値計算のための技術開発
英文：Development of Parallel Numerical Computation Technology for Post-Peta Era

須田礼仁
Reiji Suda

東京大学 情報理工学系研究科
Graduate School of Information Science and Technology, the University of Tokyo
http://sudalab.is.s.u-tokyo.ac.jp/~reiji/

邦文抄録
我々は通信を削減した疎行列ベクトル積である行列冪カーネルについて、通信の回数は複数になるが、一般の非零構造を持つ疎行列に対して計算重複の少ない手法を提案した。小規模行列で評価した結果、実際に計算重複が効果的に削減される行列がある一方、通信削減が有効でない行列が存在することも示した。

英文抄録（100 words 程度）
We proposed a method to reduce duplicated computations in matrix powers kernel for matrices with general non-zero patterns. In the experiments, our method actually reduces the duplicated computations. We also notice that there are matrices to which matrix powers kernel is not effective.

*Keywords:* 5つ程度
通信削減アルゴリズム、行列冪カーネル、計算重複の削減

## 背景と目的

半導体技術の進歩により、プロセッサ性能が高くなっているため、光の速度で制限を受ける通信時間は相対的に長く見えるようになっている。とりわけ、きわめて多数のプロセッサをネットワークで結合したスーパーコンピュータにおいて、通信時間の影響が大きい。場合によっては、処理時間の半分以上が通信の時間であるということも珍しくなくなっている。このような条件下では、多少の計算量を増やしてでも通信の量や回数を削減する通信削減アルゴリズム（Communication-Avoiding algorithms）が有効であると期待されており、様々な研究が行われている。

疎行列を係数とする連立一次方程式の反復解法などにおいて通信削減アルゴリズムを設計する際、ひとつの構成要素となるのが行列冪カーネル（Matrix Powers Kernel: MPK）である。偏微分方程式を有限差分法などで離散化して得られる疎行列の場合、非零構造に規則性があり、ステンシル計算で用いられている時空間ブロッキング（temporal blocking）の手法を使って行列冪カーネルが実現できる。しかし有限要素法などにより不規則な非零構造を持つ行列に対して時空間ブロッキングの手法を拡張することは自明ではない。

一般の非零構造を持つ行列に対する行列冪カーネルとして、従来研究では、Hoemmen の論文に含まれる PA1, PA2 という手法が知られている。これらの手法では通信の回数は（集合的に）1 回だけとなるが、計算の重複が大量に発生するという課題があった。これに対し、我々は計算の重複がほとんどない手法を提案した。ただし、我々の手法では通信の回数は複数回になってしまう。すなわち、我々の手法を従来手法とともに選択肢とすることにより、計算コストと通信コストのトレードオフを取ることができる。

## 概要

正方行列 $A$ とベクトル $x$ に対し、$x^{(k)} = A^k x$ を $k = 1,2,\dots,K$ に対してすべて求める計算が行列冪カーネルである。ここで $K$ は正の整数である。途中結果を残したい場合と、最終結果 $x^{(k)}$ だけがほしい場合とがあるが、計算とプロセッサ間通信については同じと考えてよい。これらの計算をできるだけ少ない通信回数で実行したい。本稿では行列 $A$ は対称で、対角要素はすべて非零であるとする。またベクトル $x^{(k)}$ の第 $i$ 要素を $x_i^{(k)}$ と表す。

添え字の集合は重なりのない部分集合に分割され、プロセッサに割り当てられているとする。通常の用語に従い、各プロセッサに割り当てられている部分集合を領域と呼ぶ。あるプロセッサがベクトル $x = x^{(0)}$ のうち領域 $I$ の要素 $x_j^{(0)} | j \in I$ を持っているとする。このとき、添え字 $i$ に対して $\{j \,|A_{ij} \neq 0\} \subseteq I$ であれば、$x_i^{(1)}$ は他のプロセッサと通信することなく、手元のデータだけで計算できる。このように通信なしで計算できる $x^{(1)}$ の要素をすべて計算したとする。すると、$x^{(2)}$ の一部がそれらの要素だけを参照して計算できる可能性がある。このようにして、手元のデータだけから計算できる限りの要素の集合を「錘」と呼ぶ。ステンシル計算の場合、領域を底辺とする錘になるからである。

このようにして、各プロセッサが通信なしに計算できる要素をできるかぎり計算する。これを第 0 ステップの計算と呼ぶ。次に、新たな領域分割 $I^{(1)}$ を定義する。各プロセッサは新たな領域とその袖領域に含まれる第 0 ステップの計算結果を受信する。これが第 1 ステップの通信となる。次に、先と同様に、手元のデータから計算できる限りの要素を計算する。これが第 1 ステップの計算となる。第 1 ステップの計算が終わると、さらに新たな領域分割 $I^{(2)}$ を定義し、通信と計算を行う。これを繰り返すことにより、計算が進んでゆく。

各ステップで計算がどこまで進むかを考える。大雑把にいうと、領域の端ではあまり計算が進まず、領域の中央部では計算が進むことがわかる。このため、いずれかの添え字が常に領域の端の方にあると、そこだけ計算が進まないことになる。これを避けるため、計算が進んでいるところが領域の端になるように領域分割を定義する必要がある。そのような機能は通常のグラフ分割ソフトウェアに備わっている。我々は METIS を用い、辺に重みをつけて領域分割を実施した。すなわち、計算が進んでいる頂点を結ぶ辺の重みは軽く、計算が進んでいない頂点を結ぶ辺の重みを重くすることで、計算が進んでいない頂点が領域の中央部に来やすいようにすることができた。

このような領域分割を何度も繰り返すのはコストがかかる。そこで我々は冪指数 $K$ と通信の回数 $C$ を固定して、同一の計算と通信が繰り返されるようにアルゴリズムを設計することとした。具体的には、最後の領域分割 $I^{(C)}$ は最初の領域分割 $I^{(0)}$ と同じとし、最後のステップの通信と計算は前述の方式によらず、Hoemenn の論文の PA2 と同様に、各プロセッサが担当領域の最終結果 $x_i^{(K)} | i \in I^{(C)}$ を計算するように、必要な通信と計算を行うものとした。これにより、$K$ 段までの計算が終了するとデータ分散が初期状態に戻り、同じ通信・計算パターンで行列・ベクトル積を繰り返し計算することができるようになる。

## 結果および考察

我々はまだ並列実装には至っていないので、小規模行列を用いて提案手法の計算量と通信回数を評価した。$100 \times 100$ の 2 次元メッシュを 25 プロセッサに分割し、$K = 10$ まで計算させた例では、従来手法 PA1 は 84%、PA2 は 52% 計算量が増加したのに対し、提案手法では $C = 2$ で 14%、$C = 3$ で 1% の計算量の増加で済んだ。なお PA2 は提案手法の $C = 1$ と同一である。3 次元メッシュや Florida 疎行列コレクションの shuttle_eddy でも同様の傾向が得られた。これに対し、Florida 行列の cage10 では、従来手法 PA1 では計算量が 15.3 倍、PA2 では 14.8 倍であり、提案手法でも $C = 5$ でも 5.4 倍の計算量となった。この条件では、多くのアーキテクチャでは、行列ベクトル積のたびに通信を行う古典的な手法のほうが有効ではないかと推測される。このように、行列冪カーネルの有効性が行列の非零構造に依存することも明らかとなった。

## まとめ、今後の課題

本報告では、一般的な非零構造を持つ行列に対する行列冪カーネルにおいて計算の重複を削減する手法を報告した。実際にいくつかの行列に適用し、計算の重複の削減が達成されていることを確認した。

通信削減反復法の並列実装は行っているが、今回提案した手法はまだ並列実装されておらず、それが第一の課題である。また、提案手法は通信遅延隠蔽ができないので、そのようなアルゴリズムの拡張などが望まれる。