5G-06

大規模なカルシウムイメージングデータに対する自動ソーティングアルゴリズムのチューニング

根本　貴大†　　田中　輝雄†　　藤井　昭宏†　　竹川　高志†

工学院大学†

1. はじめに

近年，神経回路に対する解像度の高い大規模なカルシウムイメージングの記録が可能となった[1]．それに伴い，記録データからスパイク列を自動的に抽出する自動ソーティングアルゴリズムが提案されている．そのうちのひとつの手法として hotaru アルゴリズム[2]が竹川らによって提案された．このアルゴリズムは確率モデルと MAP 推定および逐次二次計画法を用いて解となるスパイク列を求める手法であり，大規模な線形方程式を反復して解くことが必要となる．

我々はこの計算を高速化するために，線形方程式の形状に合わせた BLAS の並列呼び出しの実装を行ってきた[3]．一方で，BLAS の適応が困難な部分の高速化や高並列環境での高速化が課題である．また，高並列化の進歩により NUMA（Non-Uniform Memory Access）などの構成が一般的になっている．

本研究では高並列環境のひとつである NUMA におけるマルチスレッドに適したデータ配置，および帯行列ベクトル積（BMV）において行方向にブロッキングについて報告する．

2. hotaru アルゴリズム

カルシウムイメージングにより得られた観測データには多数のニューロンのカルシウム濃度の変化が動画により記録されている．図１は観測データの例である．hotaru アルゴリズムは記録データのニューロンの位置・形状および発火によるカルシウム濃度の変化を確率モデルにし，MAP 推定により逐次二次計画問題として解を求めることにより個々のニューロンの発火タイミングであるスパイク列を求める．逐次二次計画問題の解法には主双対内点法を利用し，解の探索方向の算出に線形方程式が用いられている．この線形方程式は個々のニューロンの全ての時間および全ての空間に未知数を持つため，大規模な問題サイズとなる．hotaru アルゴリズムでは線形問題の格納形式および計算手法を工夫し，問題サイズの課題を解消している[3].

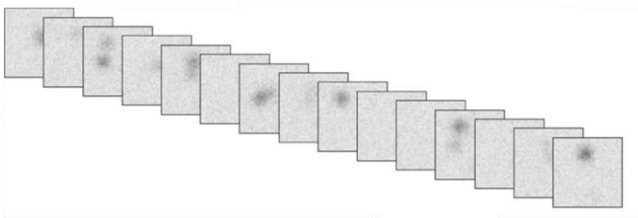

図１　カルシウムイメージングデータの例

A Performance Tuning of Automatically Sorting System for Large Scale Calcium Imaging Data
Takahiro Nemoto†, Teruo Tanaka†, Akihiro fujii†, Takashi Takekawa†
†Kogakuin University

hotaru アルゴリズムでの線形解法の計算には行列積や行列ベクトル積などの一般的な線形計算があり．これまで行列積の並列化を行った[3].

2.1 帯行列ベクトル積

hotaru アルゴリズムでは確率モデルのうち，ニューロンの発火タイミングである時間パラメータとニューロンの活動領域である空間パラメータが推定対象となる．時間パラメータはカルシウム濃度であるが，ニューロンの発火によりカルシウムが変動するというモデルを拘束条件として用いているため，主双対内点法の線形解法においてもこの特徴を含む線形方程式を解く必要がある.この計算は図２に示す斜線上の要素a,b,cがすべて同じの三重帯行列$\boldsymbol{A}$とその転置行列$\boldsymbol{A^t}$，個別の対角成分dを持つ行列$\boldsymbol{D}$とベクトル$\boldsymbol{x}$の積により計算される．従来研究はこの帯行列ベクトル積をスカラベクトル積とベクトルの内積に置き換え．計算を行っていた．そのため$\boldsymbol{x}$を7 回逐次に参照することとなり，キャッシュのヒット率が低く，計算時間増大および高並列の台数効果が得られない原因であったと考えられる．

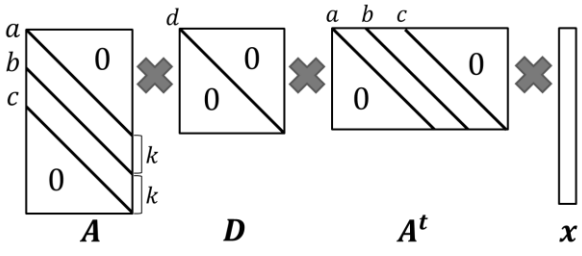

図２　帯行列ベクトル積

3. 帯行列ベクトル積の高速化

本研究では帯行列ベクトル積の計算部分に対し，二点のコードチューニングを行った．第一にキャッシュのヒット率を向上させる行分割，第二に大規模なマルチスレッド環境に適したデータ配置である．

3.1 行分割

帯行列ベクトルの各斜線要素は一定の幅kで離れている．そのため，この幅の単位を利用した行分割を行うことにより，計算処理をk行ごとに区分することができる．この計算処理ごとにスレッドを割り当てることにより，データの再利用による，キャッシュのヒット率の向上を図る．

3.2 マルチスレッドに適したデータ配置

NUMA 構成の高並列環境のスレッド並列実行では，各スレッドが使用するデータのメモリ上での位置が実行時間に影響する．

従来の研究では OpenMP による並列化は計算部分へのディレクティブの記入のみであった．この実装では NUMA で実行した場合メモリ上のデータ配置によって読み込みにロスが発生する．

本研究では 線形解法に使用する問題行列およびベクトルのデータ初期化を，計算時の分割と一致するように並列して行った．このデータ初期化の並列化により、各スレッド が必要とするデータ を自スレッドに近いメモリに配置することとなり、メモリ競合を抑えることができる．

4. 数値実験

帯ベクトル積の計算の実装において従来の行列ベクトル積の計算，行分割ごとに行う計算，データ初期化の並列化と行分割の計算の 3 種類を実装し，計算時間の比較を行った．今回は hotaru アルゴリズムで利用される線形解法（CG 法）を抽出し，100 回反復を行い 1 反復の時間計測した．行分割のサイズは 200，問題サイズ N は 20,000,000 とした．実行環境は表 1 に示すものとなっている．実装には OpenMP の並列化を適応し，並列数は最大で 20 とした．これは実行環境の CPU のコア数と同じである．

図 3 は帯行列ベクトル積の各計算時間の比較である．スレッド数 20 のとき従来に比べ行分割で 24%，行分割とデータ初期化の並列化は 18%の計算時間となった．また図 4 は線形解法の全体の時間のうち，帯行列ベクトル積の計算時間の内訳を示したものである．行分割とデータ初期化の並列化により従来の 51%の計算時間となった．BMV の計算時間は当初全体の 31%である

表 1　実験環境

CPU	IntelXeon E5-2650 v3 @2.30GHz
コア数	10 [core]×2 [CPU]
コンパイラ	icc
ライブラリ	OpenMP, Intel MKL

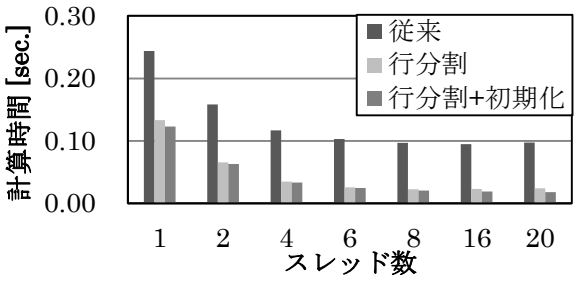

図 3　帯行列ベクトル積の実行時間の比較

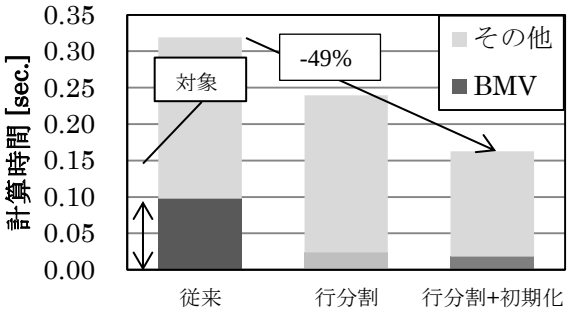

図 4　プログラム全体における実行時間の比較
（20 スレッドの場合）

が全体の時間削減は 18%多い．これはデータの初期化の並列化により BMV 以外の処理に対しても適したメモリ配置となったためと考えられる．

5. おわりに

本研究では hotaru アルゴリズムにおける線形解法の計算において帯行列積の問題構造に着目したチューニングを行うことにより，従来の行列積を逐次的に行う手法と比べ台数効果が向上および計算時間の削減を行った．

今回の実験では疑似的な線形解法により計測を行ったため，hotaru アルゴリズム全体で同様の削減効果が得られるかの検証が課題である．

参考文献

[1] M. Sato, et. al, “Generation and Imaging of Transgenic Mice that Express G-CaMP7 under a Tetracycline Response Element”, Plos One 10, e0125354(2015).

[2] T. Takekawa, et. al, "Sparse MAP inference of cell shapes and spike trains from calcium imaging data with non-negativeconstraints", Annual meeting of Society for Neuroscience(2015).

[3] 根本 貴大, et. al ,“大規模なカルシウムイメージングデータに対する自動ソーティングアルゴリズムの高速化,”Fit 15, B-011 (2016).