

## 星間化学シミュレーションにおける最適な疎行列ベクトル積演算手法

本山 一隆<sup>1,\*</sup>, 合田 憲人<sup>1</sup>, 坂根 栄作<sup>1</sup>, 西村 健<sup>1,2</sup>, 佐賀 一繁<sup>1,3</sup> (\* motoyama@nii.ac.jp)

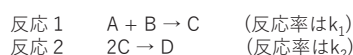
1. 国立情報学研究所、2. 総合研究大学院大学、3. 富士通株式会社

### abstract

星間ガスの中では、生命の誕生に関わるような有機分子を含め、様々な分子が生成されていることが明らかになってきている。星間化学シミュレーションの高速化は、星間ガス中での化学進化を明らかにするために必要不可欠である。本研究では、星間化学シミュレーションにとって最適な疎行列ベクトル積演算手法を明らかにするため、3種類の演算手法、Coordinate Storage(COO)形式、Compressed Column Storage(CCS)形式、Compressed Row Storage(CRS)形式を実装し、その性能比較を行った。実験の結果、逐次計算と並列計算のいずれの場合もCRS形式が最も高い性能を示した。また、疎行列演算ライブラリ Sparse BLASを用いた計算では、単純なOpenMPの実装よりも性能が低かった。

### 星間化学シミュレーションにおける疎行列演算

例として、分子A, B, C, Dが、次の化学反応を起こすとする。



このとき、それぞれの分子の濃度( $n_x$ )の時間変化は

$$\frac{d}{dt} \begin{pmatrix} n_A \\ n_B \\ n_C \\ n_D \end{pmatrix} = \begin{pmatrix} -1 & 0 \\ -1 & 0 \\ 1 & -2 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} k_1 \\ k_2 \end{pmatrix}$$

化学量論行列      反応率ベクトル

と表せる。星間化学シミュレーションの性能は、化学量論行列(化学反応式の係数行列)と反応率ベクトルの積の演算効率で決まる。

下図は、星間化学データベース(<http://udfa.ajmarkwick.net>)に登録されている化学反応から作った化学量論行列



- 行数は468(化学種の数)、列数は6175(化学反応式の数)
- 非零要素が全要素中の0.85%(24580個)しかない疎行列

疎行列演算の効率は非零要素の分布の仕方に依存しており、最適な演算手法は行列ごとに違っている。

### 研究の目的

星間化学シミュレーションに最適な疎行列ベクトル積の演算手法を明らかにする

### 疎行列演算手法

#### COO形式

- 非零要素を任意の順でAに保存
- 非零要素の列番号と行番号を col\_ind, row\_indに保存

下の計算を例として、COO形式、CCS形式、CRS形式の演算手法を示す。

$$\begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 2 \\ 0 & 0 & 3 & 0 \\ 0 & 4 & 5 & 0 \\ 0 & 0 & 6 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix}$$

```
A = [1, 2, 3, 4, 5, 6]
col_ind = [1, 4, 3, 2, 3, 3]
row_ind = [1, 1, 2, 3, 3, 4]
```

```
do k = 1, n_nz
  ii = row_ind(k)
  jj = col_ind(k)
  y(ii) = y(ii) + A(k) * x(jj)
end do
```

#### CCS形式

- 非零要素を列方向にAに保存
- 非零要素の行番号を row\_ind, 各行最初の非零要素のA内での位置を col\_ptrに保存

```
A = [1, 4, 3, 5, 6, 2])
row_ind = [1, 3, 2, 3, 4, 1]
col_ptr = [1, 2, 3, 6]
```

```
do j = 1, n_col
  do i = col_ptr(j), col_ptr(j+1)-1
    ii = row_ind(i)
    y(ii) = y(ii) + A(i) * x(j)
  end do
end do
```

#### CRS形式

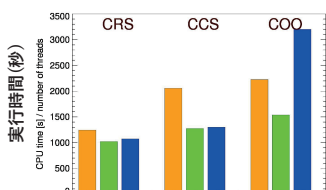
- 非零要素を行方向にAに保存
- 非零要素の列番号を col\_ind, 各行最初の非零要素のA内での位置を row\_ptrに保存

```
A = [1, 2, 3, 4, 5, 6]
col_ind = [1, 4, 3, 2, 3, 3]
row_ptr = [1, 3, 4, 6]
```

```
do i = 1, n_row
  do j = row_ptr(i), row_ptr(i+1)-1
    jj = col_ind(j)
    y(i) = y(i) + A(j) * x(jj)
  end do
end do
```

### 性能比較

Rollig et al. 2007のベンチマーク問題の実行にかかった時間を比較した。



オレンジ: 逐次計算  
緑: OpenMPによる並列計算(4スレッド)  
青: SBLASを用いた並列計算(4スレッド)

- 逐次計算と並列計算のいずれでも、**CRS形式を用いた場合が最も性能が高い。**
- Sparse BLASを用いた場合は、単純なOpenMPによる並列計算より性能が低かった。
- どの形式でも並列化効率は低かった。原因として、配列の間接参照によるキャッシュミスの発生が考えられる。

#### 実験環境

|               |                        |
|---------------|------------------------|
| CPU           | Intel Core i5 2.66 GHz |
| 主記憶           | DDR3 4 GB              |
| L1/L2/L3キャッシュ | 32 KB/256 KB/8 MB      |

### 今後の課題

- 他の疎行列演算手法の調査  
今回扱った代表的な3種類の演算手法以外にも、性能の調査が必要である。
- キャッシュヒット率の改善  
行列の行や列の順番を並び替えることによって非零要素を局在化させ、キャッシュヒット率を改善する。