

反応拡散方程式の数値計算並列処理

- 陽的差分法を用いた反応拡散方程式の数値解
 - 1次元の場合、空間の分割数 N に比例した計算量
 - 2次元の場合、N² に比例した計算量
 - 空間を分割して複数ノードを用いた並列処理は効果的か？

1次元反応拡散方程式

$$\frac{\partial u}{\partial t} = D_u \frac{\partial^2 u}{\partial x^2} + f(u, v) \quad \text{Schnakenberg kinetics}$$

$$f(u, v) = k_1 - k_2 u + k_3 u^2 v$$

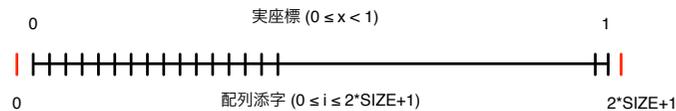
$$\frac{\partial v}{\partial t} = D_v \frac{\partial^2 v}{\partial x^2} + g(u, v) \quad g(u, v) = k_4 - k_3 u^2 v$$

区間 $0 \leq x \leq 100$ を

- N = 10,000 で分割して 1ノードで計算
- 2ノードを用いて (N = 5,000) 並列計算

陽的差分法

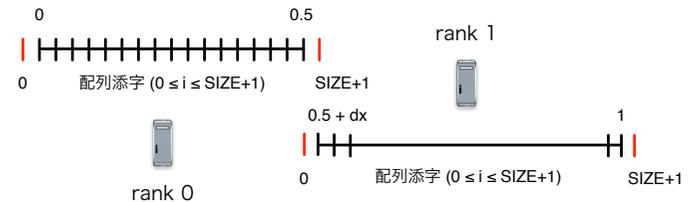
1ノードで実行する場合 (これまでのやり方)



```
double u[2*SIZE+2], v[2*SIZE+2];   u[0], u[2*SIZE+1] は境界条件用のダミー
void go_ahead() /* 時刻を dt だけ進める */
{
  for(i=1; i<2*SIZE+1; i++){
    ud[i] = u[i] + .....
  }
}
```

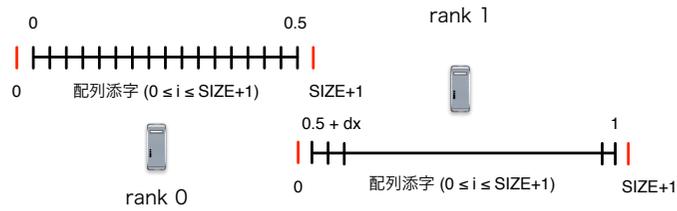
2ノードによる並列処理

2ノードで実行する場合 (空間を2つに分割)



2つに分割した空間をうまく接続する必要がある
(rank 0 と 1 の間の通信)

2ノードによる並列処理実装



```
double u[SIZE+2], v[SIZE+2];  
  
void go_ahead(int rank)  
{  
    MPIにより必要な境界条件を取得 (send/receive)  
    for(i=1; i<SIZE+1; i++){  
        ud[i] = u[i] + .....  
    }  
}
```

2ノードによる並列処理実装

- 時刻を dt 進める関数 process(int rank) を定義
- rank 0 は左半分の区画、rank 1 は右半分の区画を担当

```
void go_ahead(int rank)  
{  
    if( rank == 0 ) {  
        MPIにより右端の境界条件を取得 (send/receive)  
        陽的差分  
    } else if( rank == 1 ){  
        MPIにより左端の境界条件を取得 (send/receive)  
        陽的差分  
    }  
}
```

2ノードによる並列処理実装

- それぞれのノードの計算結果を、個別のファイルに書き出す。
- Mathematica で2つのファイルを連結して図示。

並列処理の効果

- 時刻を dt 進めるために必要な計算量は分割数 N に比例
- ノード間通信に要する時間 T_c (通信には時間がかかる)
- 陽的差分の処理時間 + 通信時間が実時間

1ノード処理

2ノード処理

$$T \propto 2N$$

$$T \propto N + T_c$$

分割数 N が大きいほど 2ノード並列処理が効果的

課題

- Schnakenberg kinetics による1次元反応拡散方程式を2つのノードを用いて並列計算するプログラムを作成せよ。
- 処理速度（実行時間）を測定し、並列化の効果（2ノードを用いると2倍速くなるのか？）について調べよ。分割数 N を変化させ得ると並列化の効果はどうか？
- パラメータは次の値を用いること。

$$k_1 = 0.2, k_2 = 1.0, k_3 = 1.0, k_4 = 0.5$$

$$D_u = 1.0, D_v = 14.0$$

$$0 \leq x \leq 100$$