KOBE HPC サマースクール 2024（初級）

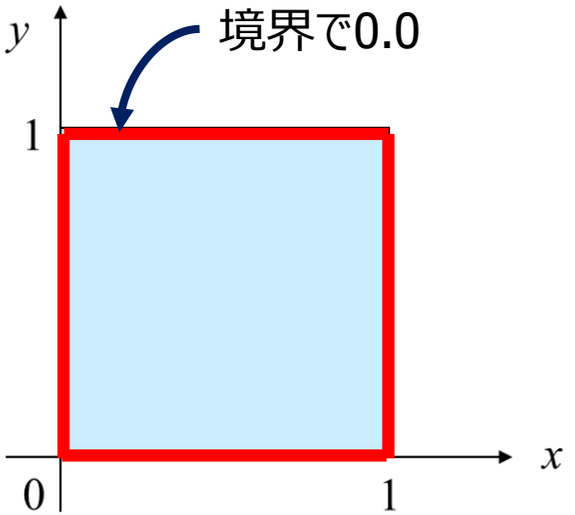
12. 熱伝導問題の並列計算

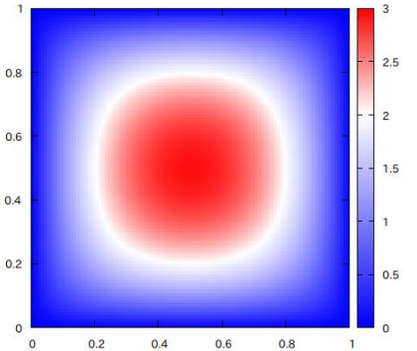
13. ハイブリッド並列

2次元熱伝導問題

- 2次元正方形領域 [0,1]×[0,1] に一定の熱を加え続けた時の領域の温度がどうなるか？

$$\frac{\partial u}{\partial t} = \Delta u + 40 = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + 40 \to 0$$





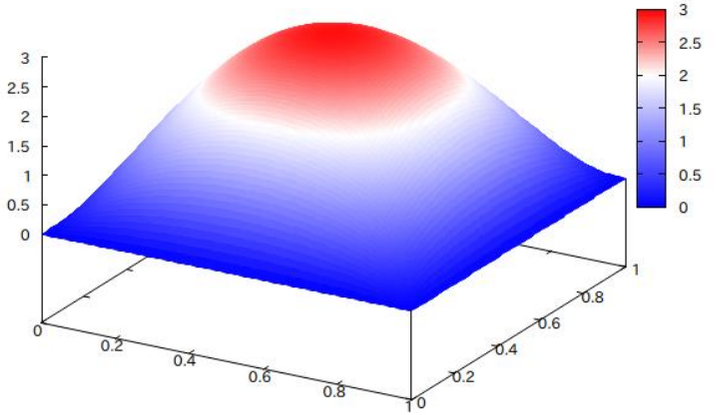
- 境界条件

$$u(0, y) = 0.0 \quad (0 \le y \le 1)$$

$$u(1, y) = 0.0 \quad (0 \le y \le 1)$$

$$u(x, 0) = 0.0 \quad (0 \le x \le 1)$$

$$u(x, 1) = 0.0 \quad (0 \le x \le 1)$$



熱伝導方程式の差分表現

- 空間（xおよびy方向）2階微分 ⇒ 中心差分

$$\frac{\partial^2 u}{\partial x^2} \sim \frac{u_{i+1,j}^t - 2u_{i,j}^t + u_{i-1,j}^t}{(\Delta x)^2}, \qquad \frac{\partial^2 u}{\partial y^2} \sim \frac{u_{i,j+1}^t - 2u_{i,j}^t + u_{i,j-1}^t}{(\Delta y)^2}$$

- 時間微分 ⇒ 前進差分

$$\frac{\partial u}{\partial t} \sim \frac{u_{i,j}^{t+\Delta t} - u_{i,j}^t}{\Delta t}$$

- 熱伝導方程式の差分表現

$$\frac{u_{i,j}^{t+\Delta t} - u_{i,j}^t}{\Delta t} = \frac{u_{i+1,j}^t - 2u_{i,j}^t + u_{i-1,j}^t}{(\Delta r)^2} + \frac{u_{i,j+1}^t - 2u_{i,j}^t + u_{i,j-1}^t}{(\Delta r)^2} + 40$$

温度の時間発展方程式の導出

- 差分表現に直した熱伝導方程式を変形

$$u_{i,j}^{t+\Delta t} = u_{i,j}^{t} + \frac{\Delta t}{(\Delta r)^2}\left(u_{i+1,j}^{t} - 2u_{i,j}^{t} + u_{i-1,j}^{t}\right) + \frac{\Delta t}{(\Delta r)^2}\left(u_{i,j+1}^{t} - 2u_{i,j}^{t} + u_{i,j-1}^{t}\right) + 40\Delta t$$

$$= \left(1 - \frac{4\Delta t}{(\Delta r)^2}\right)u_{i,j}^{t} + \frac{\Delta t}{(\Delta r)^2}\left(u_{i+1,j}^{t} + u_{i-1,j}^{t} + u_{i,j+1}^{t} + u_{i,j-1}^{t}\right) + 40\Delta t$$

$$= \alpha u_{i,j}^{t} + \beta\left(u_{i+1,j}^{t} + u_{i-1,j}^{t} + u_{i,j+1}^{t} + u_{i,j-1}^{t}\right) + 40\Delta t,$$

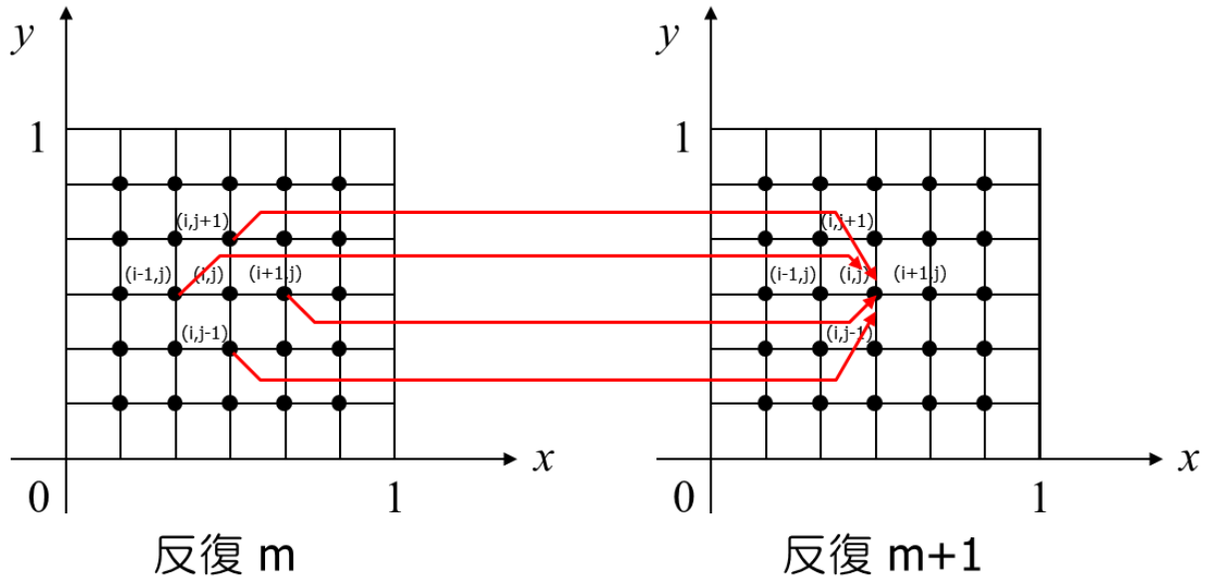
ただし，$\alpha = 1 - 4\beta$，$\beta = \frac{\Delta t}{(\Delta r)^2}$．ここで，$\beta = \frac{\Delta t}{(\Delta r)^2} = 0.25$ とすると…

$$u_{i,j}^{t+\Delta t} = 0.25\left(u_{i+1,j}^{t} + u_{i-1,j}^{t} + u_{i,j+1}^{t} + u_{i,j-1}^{t}\right) + 10(\Delta r)^2$$

有限差分法による熱伝導問題の解法

- 温度時間発展計算のアルゴリズム

```
for( i=1; i<=N; i++ ) {
  for( j=1; j<=N; j++ ) {
    u_{i,j}^{(m+1)} = (u_{i-1,j}^{(m)}+u_{i+1,j}^{(m)}+u_{i,j-1}^{(m)}+u_{i,j+1}^{(m)})*0.25 + 10.0*dx*dx;
  };
};
```



演習12a：プログラム heat2d.c の実行

- 2次元熱伝導問題の逐次プログラム heat2d.c をコンパイルして実行せよ.

```
$ cp /home/guest60/share/heat2d.c ./
$ icx heat2d.c
$ ./a.out
```

[Fortran]
```
$   cp /home/guest60/share/heat2d.f90 ./
$   ifx heat2d.f90
$   ./a.out
```

- プログラムでは $\mathrm{N} = 128$ の時に，反復ベクトルの差のノルム $\left\| u_{i,j}^{\mathrm{m}+1} - u_{i,j}^{\mathrm{m}} \right\|_2$ が閾値 $\epsilon = 1.0 \times 10^{-3}$ より小さくなったら反復を停止している.
- プログラム実行が正常に完了したら「To visualize data,...」というメッセージが出てくる．これの指示に従うと計算結果をpng画像ファイルに出力できる（gnuplot を用いる．事前に「module load gnuplot」で使用環境を load する）.

heat2d.c のMPIによる並列化（Cの場合）

■ 並列化のヒント

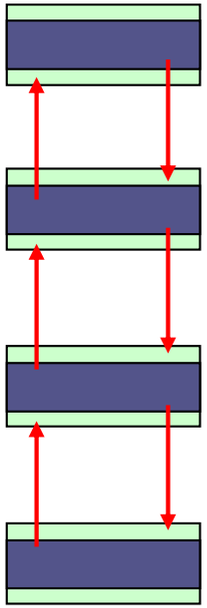
1. 2次元配列 u（u^m），u_new（u^{m+1}）をブロック行分割しておく．

2. u[is:ie][] の計算をする前に，
   - 上のプロセスの u の ie 行を下のプロセスの (is-1) 行へ，
   - 下のプロセスの u の is 行を上のプロセスの (ie+1) 行へ，
   送る．
   - 並列化 sr_matrix.c を参考にして，MPI_Sendrecv を用いて送受信．

3. u の is ～ ie 行の計算を行う．計算結果は u_new に格納．

4. 反復ベクトルの差のノルムの部分和を計算し，
   MPI_Allreduce で総和を取り，収束の判定を行うようにする．
   - ローカルで残差の部分和を計算し，各プロセスでノルムを計算する．

5. u_new を u に入れなおして，手順2に戻る．

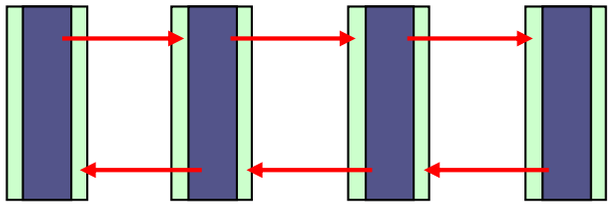


上下のプロセスから1行を受信
（受信用の領域を確保しておく）

heat2d.f90 のMPIによる並列化（Fortranの場合）

■ 並列化のヒント

1. 2次元配列 u（u^m），u_new（u^{m+1}） をブロック列分割しておく.

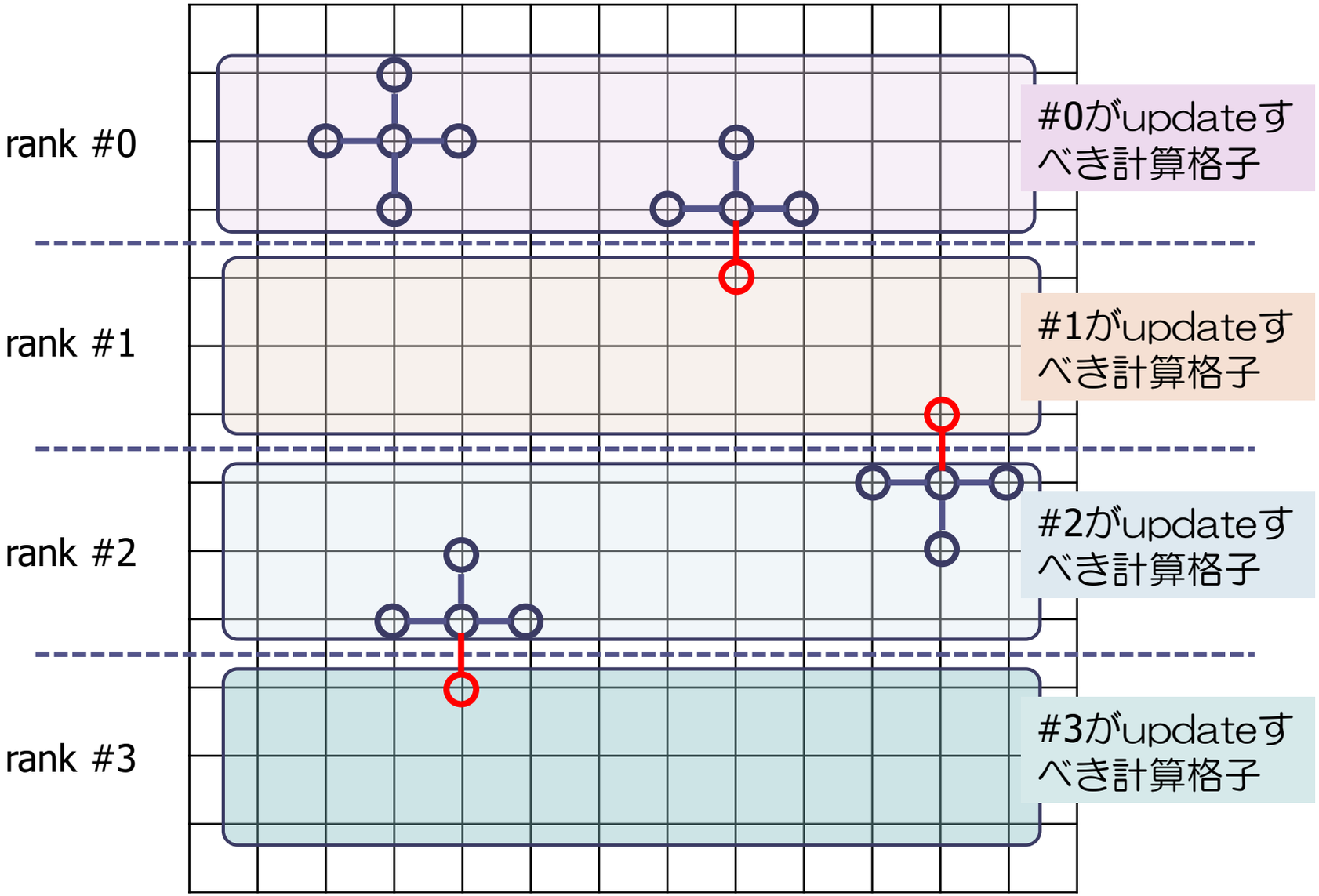


左右のプロセスから1行を受信
（受信用の領域を確保しておく）

2. U(:,js:je) の計算をする前に,
   - 左のプロセスの u の je 列を右のプロセスの (js-1) 列へ,
   - 右のプロセスの u の js 列を左のプロセスの (je+1) 列へ,
   送る.
   - 並列化 sr_matrix.f90 を参考にして，MPI_Sendrecv を用いて送受信.
3. u の js ～ je 列の計算を行う．計算結果は u_new に格納.
4. 反復ベクトルの差のノルムの部分和を計算し,
   MPI_Allreduce で総和を取り，収束の判定を行うようにする.
   - ローカルで残差の部分和を計算し，各プロセスでノルムを計算する.
5. u_new を u に入れなおして，手順2に戻る.

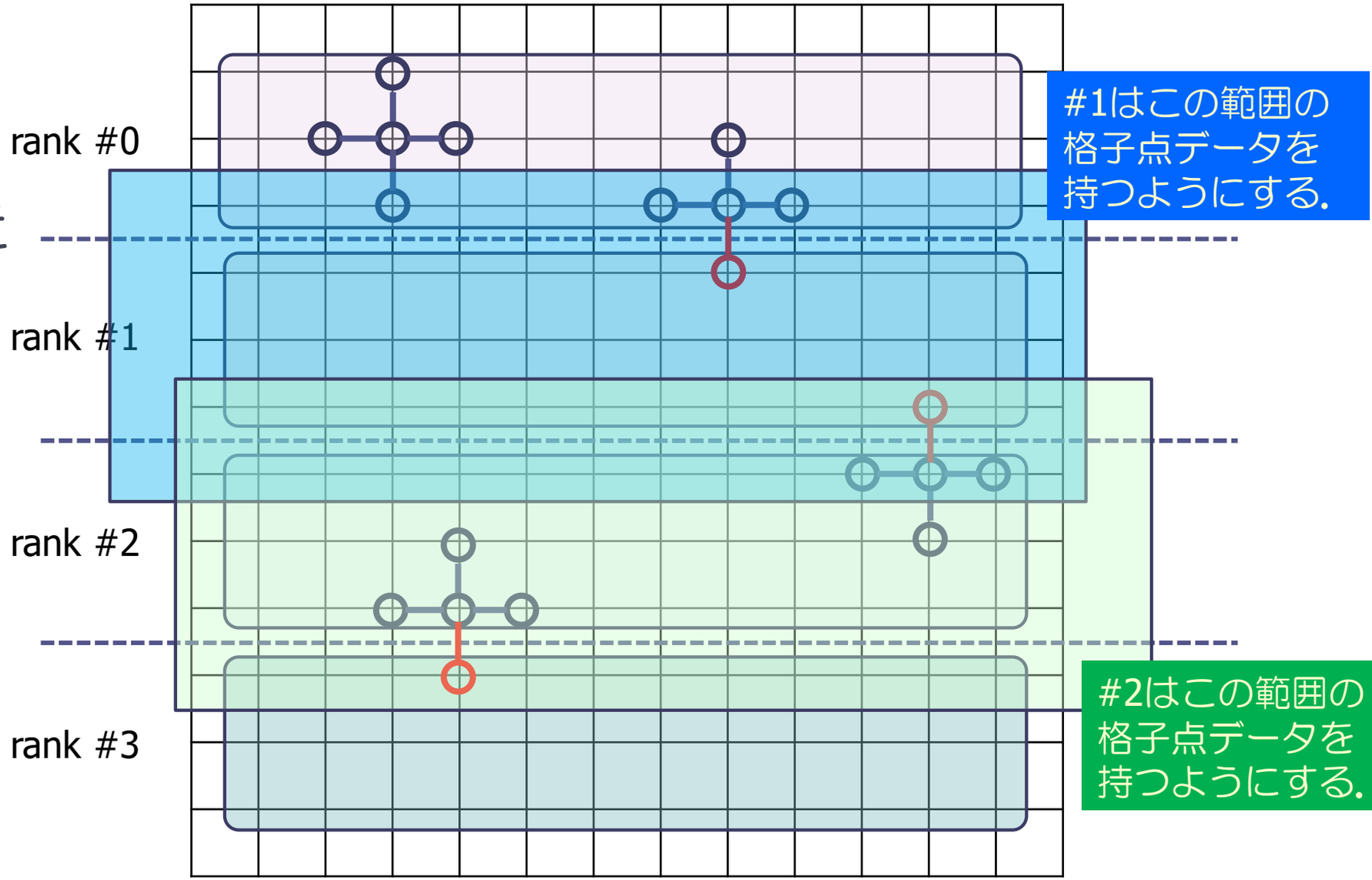
Cの場合の処理のイメージ

※Fortranの場合はこれを横に寝かしたようなイメージになる



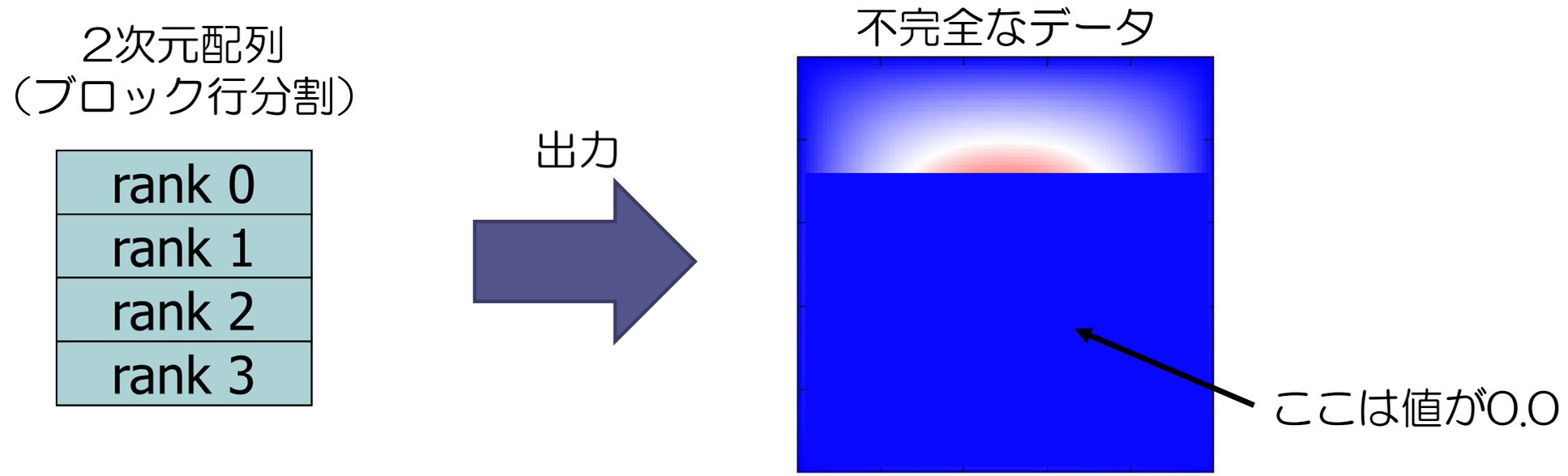
Cの場合の処理のイメージ

※Fortranの場合はこれを横に寝かしたようなイメージになる



ファイルへのデータ出力について

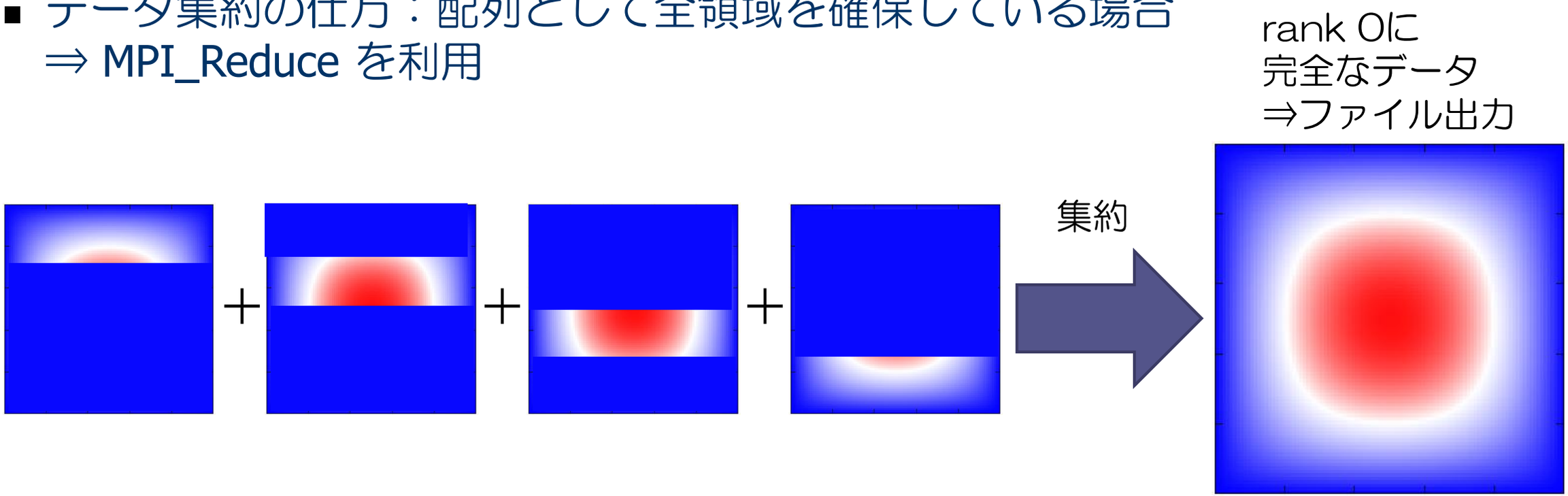
- 各プロセスは分割された小領域のみ計算を担当



- 全体領域をファイル出力するためには温度データをプロセス0に集約する必要がある.
  ⇒ プログラム中の「data output」の直前のタイミング（計3箇所）で実施

ファイルへのデータ出力について

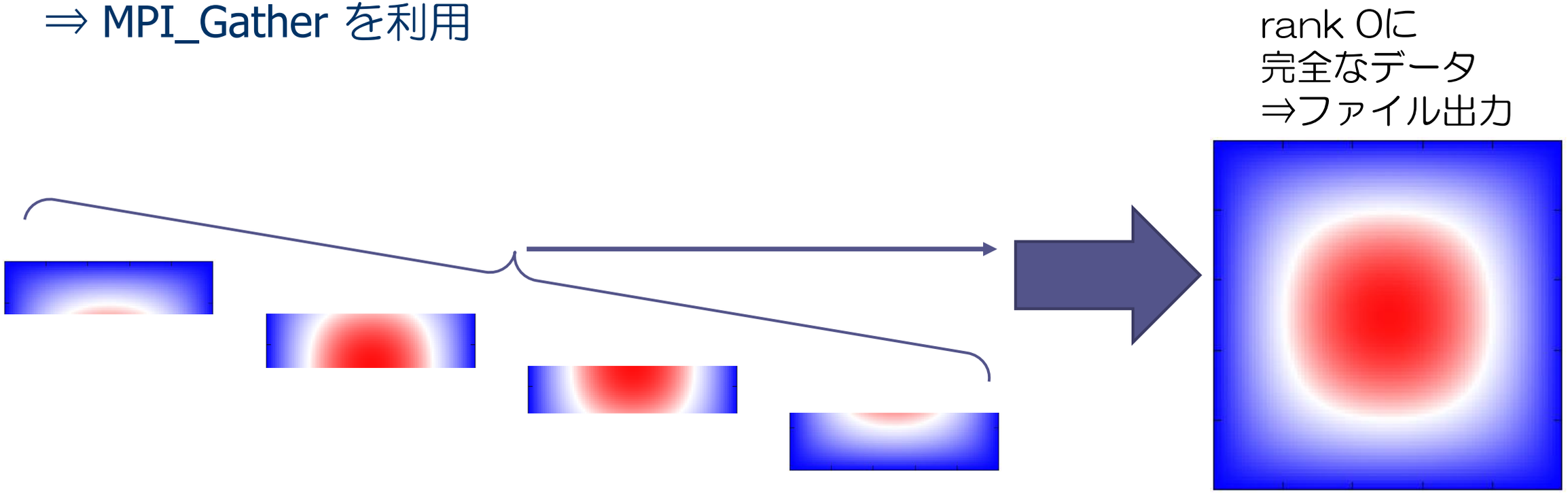
- データ集約の仕方：配列として全領域を確保している場合
  ⇒ MPI_Reduce を利用



- 注意：各プロセスのデータに重複があると，その部分の値が不正になる

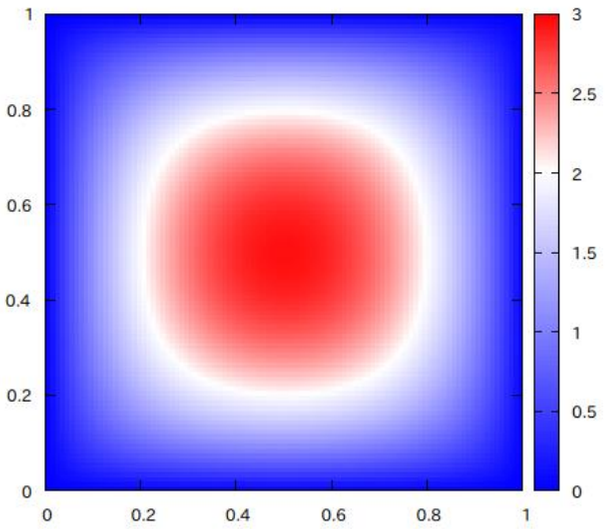
ファイルへのデータ出力について

- データ集約の仕方：配列として部分領域のみ確保している場合
  ⇒ MPI_Gather を利用



演習12b：heat2d.{c/f90} の並列化

- heat2d.c，heat2d.f90 をMPIを用いて並列化せよ.



- 計算結果を gnuplot 等で図示し，非並列版と同じ結果になることを確認せよ.
  - 計算結果の出力は，プロセス 0 にデータを集め※，ファイルに出力する.
  - 集め方（通信の仕方）は複数ある．考えてみよ.

- N=128とし，プロセス数1, 2, 4, 8 として，反復開始から終了までの計算時間を計測し，速度向上率を求めよ．また，それをグラフにせよ.
- 時間計測時は反復計算途中データのファイル出力を OFF にすると良い.
  - プログラム中のパラメータ INTV の値を 1 にするとデータのファイル出力が抑制される.
  - ファイル出力がないので，通信によるデータの収集（上述の※）も反復計算中は必要ない.

演習13：ハイブリッド並列化

- MPI を用いて並列化した heat2d.{c/f90} に，さらに OpenMP のディレクティブを挿入し，スレッド並列とプロセス並列の併用により，実行時間が削減されるこかどうか確認せよ.

  - #pragma omp parallel などを利用する（C言語）.
  - 冒頭で omp.h をインクルードするのを忘れずに（C言語）.
  - !$OMP parallel などを利用する（Fortran言語）.
  - 冒頭で !$ use omp_lib を宣言するのを忘れずに（Fortran言語）.
  - 一般にハイブリッド並列はプロセス・スレッド並列単体より高度な並列実装であるが，速度上のメリットが得られるかどうかは，計算の内容や計算機の仕様に依存する.

- コンパイル
  - `icx -qopenmp xxxxx.c -lmpi`
  - `ifx -qopenmp xxxxx.f90 -lmpi`

【サンプル】Hybrid並列プログラムの実行シェル（jobh.sh）

```
#!/bin/bash

#PBS -N hybrid
#PBS -q WS
#PBS -j oe
#PBS -l select=4:ncpus=16:mpiprocs=2

source /etc/profile.d/modules.sh
module load intel
module load mpt

export KMP_AFFINITY=disabled
export OMP_NUM_THREADS=8

cd ${PBS_O_WORKDIR}
mpiexec_mpt omplace -nt ${OMP_NUM_THREADS} ./a.out
```

ジョブ名の指定
キューの設定：スクール専用キューWS

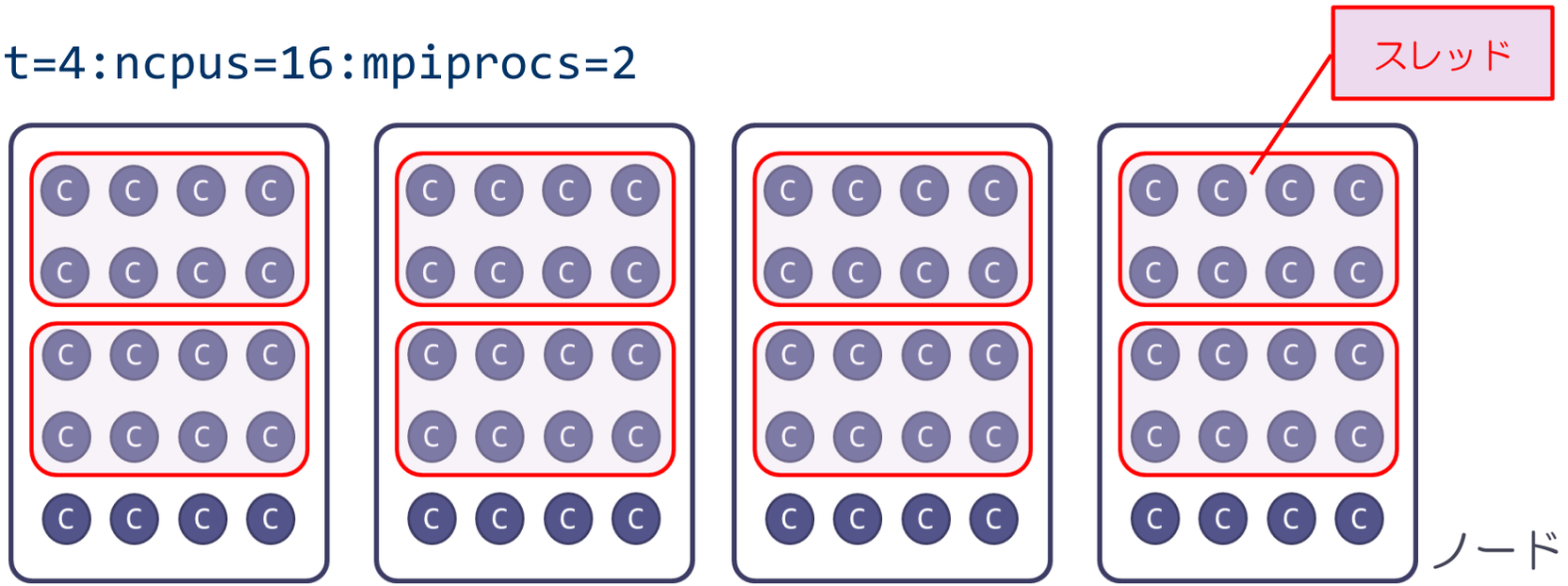
select:　計算ノード数
ncpus :　1計算ノード内のコア数
mpiprocs:　1計算ノード内のMPIプロセス数

```
#PBS -l select=N:ncpus=T:mpiprocs=M
```

- T：　1ノード（共有メモリ）のコア数（≦40）
- M：　1ノードのMPIプロセス数
- N：　全ノード数
- NM:　全MPIプロセス数

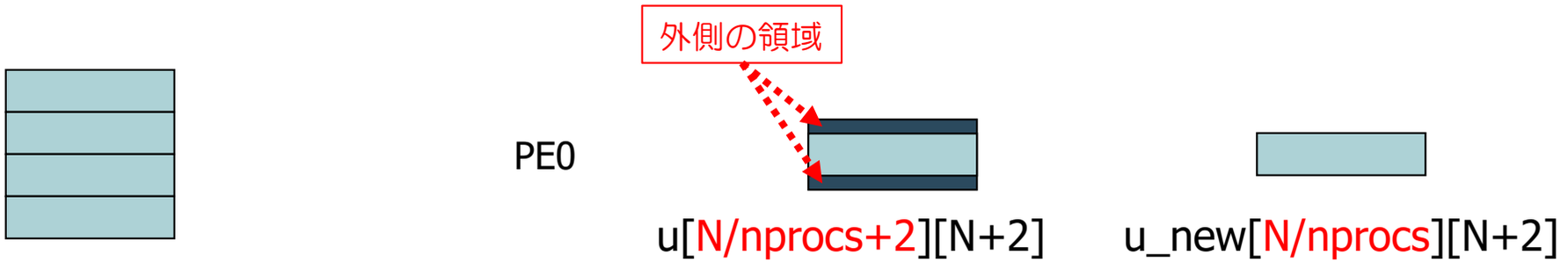
例えば，コア数を 16，全MPIプロセスを 8 とすると，以下のようなマッピングが考えられる．この場合，

```
#PBS -l select=4:ncpus=16:mpiprocs=2
```



上級者になるために．．．

- 大規模問題を解く場合には，1プロセスのメモリ使用容量を少なくする必要が出てくる．
- 今回のプログラムでは，どのプロセスも u[N+2][N+2], u_new[N+2][N+2] として，計算全領域の変数を宣言したが，



各プロセスは，これだけの大きさをもてば良いはず．

- 実行前から並列数（nprocs）が固定されていれば，上記のように，プログラムの先頭で小さい領域を宣言することも出来るが，汎用のプログラムとしては，nprocs を実行開始時に決めたい．
- 配列を動的に確保する必要がある（C言語：malloc 関数、Fortran言語：allocate文）．
  - サンプルプログラム /home/guest60/share/alloc_heat2d.c もしくは alloc_heat3d.f90