KOBE HPC スプリングスクール 2022（中級）

コミュニケータとデータタイプ（Communicator and Datatype）

2021年3月9日

神戸大学大学院システム情報学研究科計算科学専攻
横川三津夫

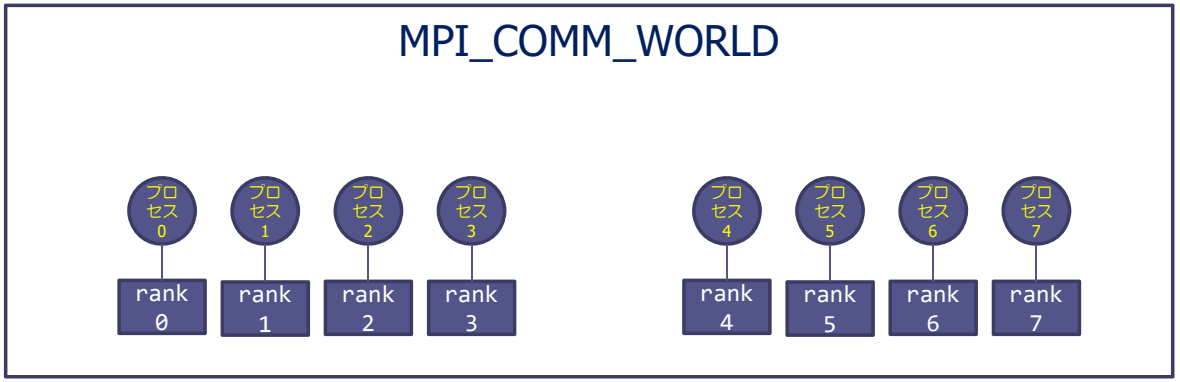
講義の内容

- コミュニケータ（Communicator）
- データタイプ（Datatype）
- コミュニケータについての演習問題

コミュニケータ（Communicator）1/2

- MPIにおけるプロセスの集団（集合）
- 集団的な操作における操作対象
  - 集団型通信（Collective Communication）：メッセージ通信がコミュニケータ内で行われる.
  - 集団型I/O（Collective Operation）
  - Window操作（One-sided Communication）

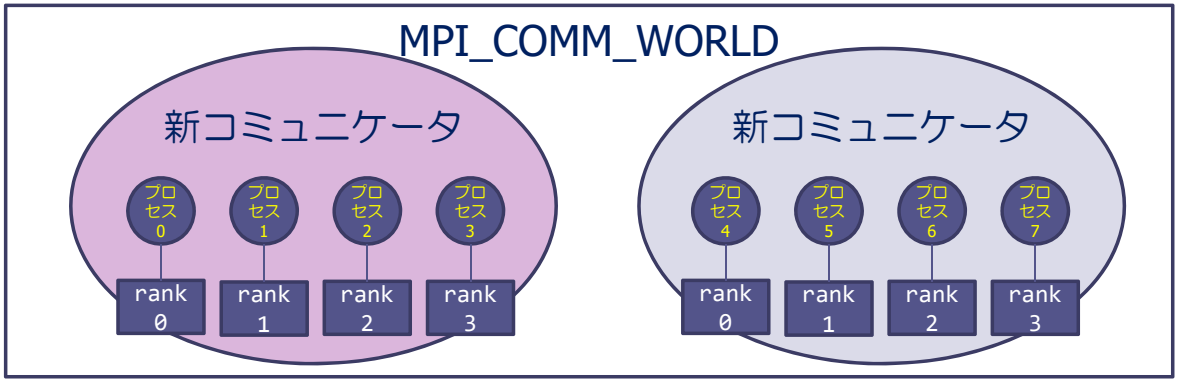
- MPI_COMM_WORLD：MPIプログラム起動後に最初に作られるプロセス全体のコミュニケータ



コミュニケータ（Communicator）2/2

- MPIにおけるプロセスの集団（集合）
- 集団的な操作における操作対象
  - 集団型通信（Collective Communication）：メッセージ通信がコミュニケータ内で行われる.
  - 集団型I/O（Collective Operation）
  - Window操作（One-sided Communication）

- コミュニケータを分割，生成：新しいランク番号が割り当てられる.



新しいコミュニケータ生成（分割）

```
【C】int MPI_Comm_split( MPI_Comm comm, int color, int key, MPI_Comm *newcomm)

【F】mpi_comm_split( comm, color, key, newcomm, ierr )
```

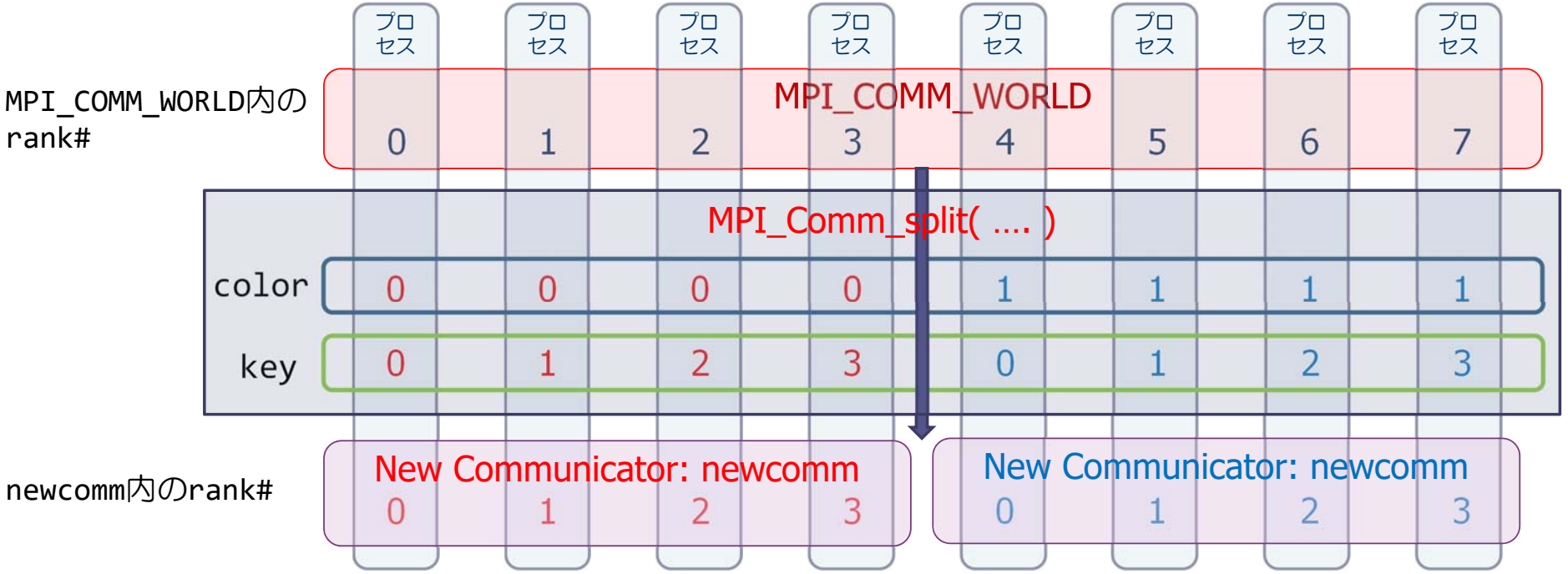
- comm: 元のコミュニケータ
- color: 生成する複数のコミュニケータを識別する変数（正の整数）
  同じ値を持つプロセスが一つのコミュニケータを構成.
- key: 同じコミュニケータ内のランク番号を指定（整数型）
  値が同じ場合には，親コミュニケータの順番が維持される.
- newcomm: 新しいコミュニケータ
  自分が属するサブグループで同じコミュニケータとなる.
- ierr: 戻りコード（整数型）

- コミュニケータを分割し新たな複数のコミュニケータを生成
- 同じcolorを持つプロセス群で（同じ名前の）新しいコミュニケータを生成
- keyの値の順番で，同一コミュニケータ内のランク順が決まる．keyの値が同じ場合は，システムが適当に決める.
- 各プロセスで同時に実行（SPMD）

例：コミュニケータを2つに分割

- MPI_COMM_WORLDを2つのコミュニケータ newcommに分割する．rank# はMPI_COMM_WORLDのランク番号．

```
color = rank# / 4;
key   = rank# % 4;
MPI_Comm_split( MPI_COMM_WORLD, color, key, &newcomm );
```



コミュニケータの複製，コミュニケータの開放

```
【C】int MPI_Comm_dup( MPI_Comm comm, MPI_Comm *newcomm );
【F】mpi_comm_dup( comm, newcomm, ierr )
```

- comm:　コミュニケータ
- newcomm:　複製のコミュニケータ
- ierr:　戻りコード（整数型）

- 親コミュニケータと同じプロセスグループの複製を生成
- 元のコミュニケータと，生成されたコミュニケータで通信を分離

```
【C】int MPI_Comm_free( MPI_Comm *newcomm );
【F】mpi_comm_free( newcomm, ierr )
```

- newcomm:　コミュニケータ
- ierr:　戻りコード（整数型）

- 指定されたコミュニケータを解放
- この関数の呼び出し以降は，当該コミュニケータは使用不能

データタイプ：MPIで扱うデータ型

- 基本データ型
  - 整数型，実数型，文字型などの基本となるデータ型
    - 【C】MPI_CHAR，MPI_INT，MPI_DOUBLEなど
    - 【F】MPI_INTEGER，MPI_REAL，MPI_DOUBLE_PRECISION

- 派生データ型（Derived Datatype）
  - 基本データ型の組合せにより，新たに生成されるデータ型
    - メモリ上の不連続なデータレイアウト（メモリ内配置）をまとめるためのデータ型
    - 不連続なレイアウトにあるデータ群を1回の通信で処理可能
  - 派生データ型生成を行う関数により生成する.
    - 基本データ型とそのオフセットの集合で生成

派生データ型の生成，登録，解放をする関数群

- 派生データ型を生成する関数（関数の一部）
  - MPI_Type_contiguous
  - MPI_Type_vector
  - MPI_Type_indexed
  - MPI_Type_create_indexed
  - MPI_Type_create_subarray
  - MPI_Type_create_darray
- 登録：生成した派生データ型を登録する関数
  - MPI_Type_commit
- 解放：不要になった派生データ型を開放する関数
  - MPI_Type_free

派生データ型の登録，解放

```
【C】int MPI_Type_commit( MPI_Datatype *type );
【F】mpi_type_commit( type, ierr )
```

- type:　作成した派生データ型
- ierr:　戻りコード（整数型）

- 新しく作成した派生データ型は，必ずこの関数を用いて登録する必要がある．
- この関数を呼び出した後は，この関数で登録したデータ型を用いた通信，I/Oなどで利用可能

```
【C】int MPI_Type_free( MPI_Datatype *type );
【F】mpi_type_free( type, ierr )
```

- type:　解放したい派生データ型
- ierr:　戻りコード（整数型）

- 必要なくなった派生データ型を解放．
- この関数を呼び出した後は，解放した派生データ型は使えない．

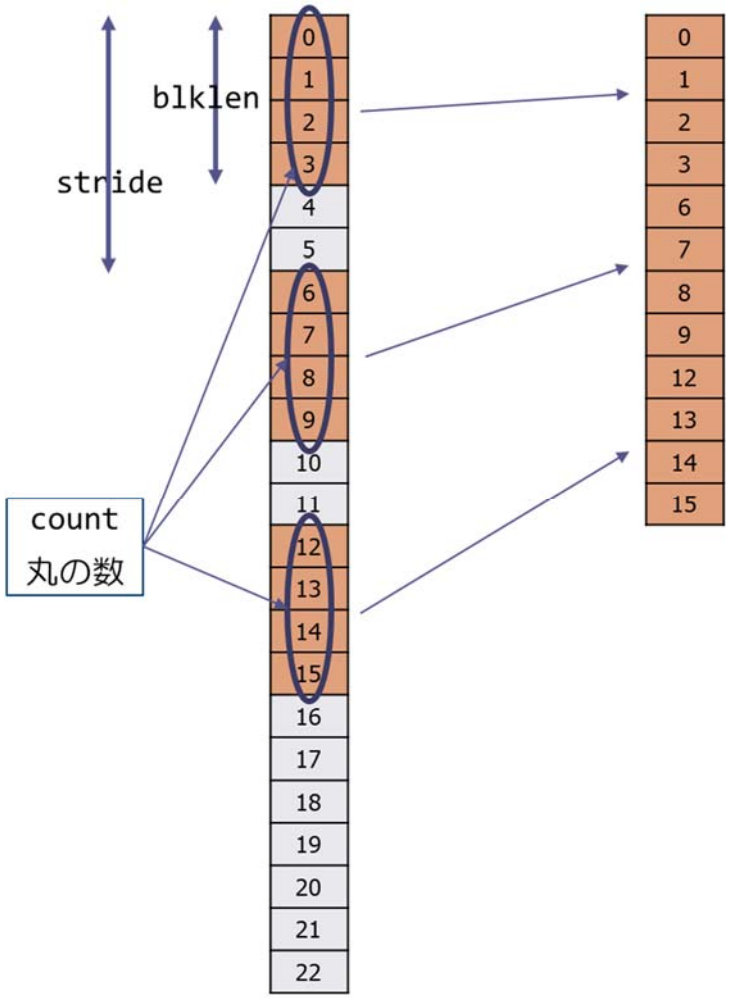
派生データ型の生成関数：MPI_Type_vector

```
【C】int MPI_Type_vector( int count, int blklen, int stride, MPI_Datatype oldtype,
                          MPI_Datatype *newtype);

【F】mpi_type_vector( count, blklen, stride, otype, ntype, ierr )
```

- count:　ブロックの数（非負整数）
- blklen:　ブロック内の要素数（非負整数）
- stride:　ブロック間の要素数（整数）
- oldtype:　元々のデータタイプ
- newtype:　新しいデータタイプ名
- ierr:　戻りコード（整数型）

【例】 1次元データの一部を転送



- 飛び飛びのデータを連続なデータとして扱えるようになる.
- 送信，受信する場合のデータ個数は 1個とする.

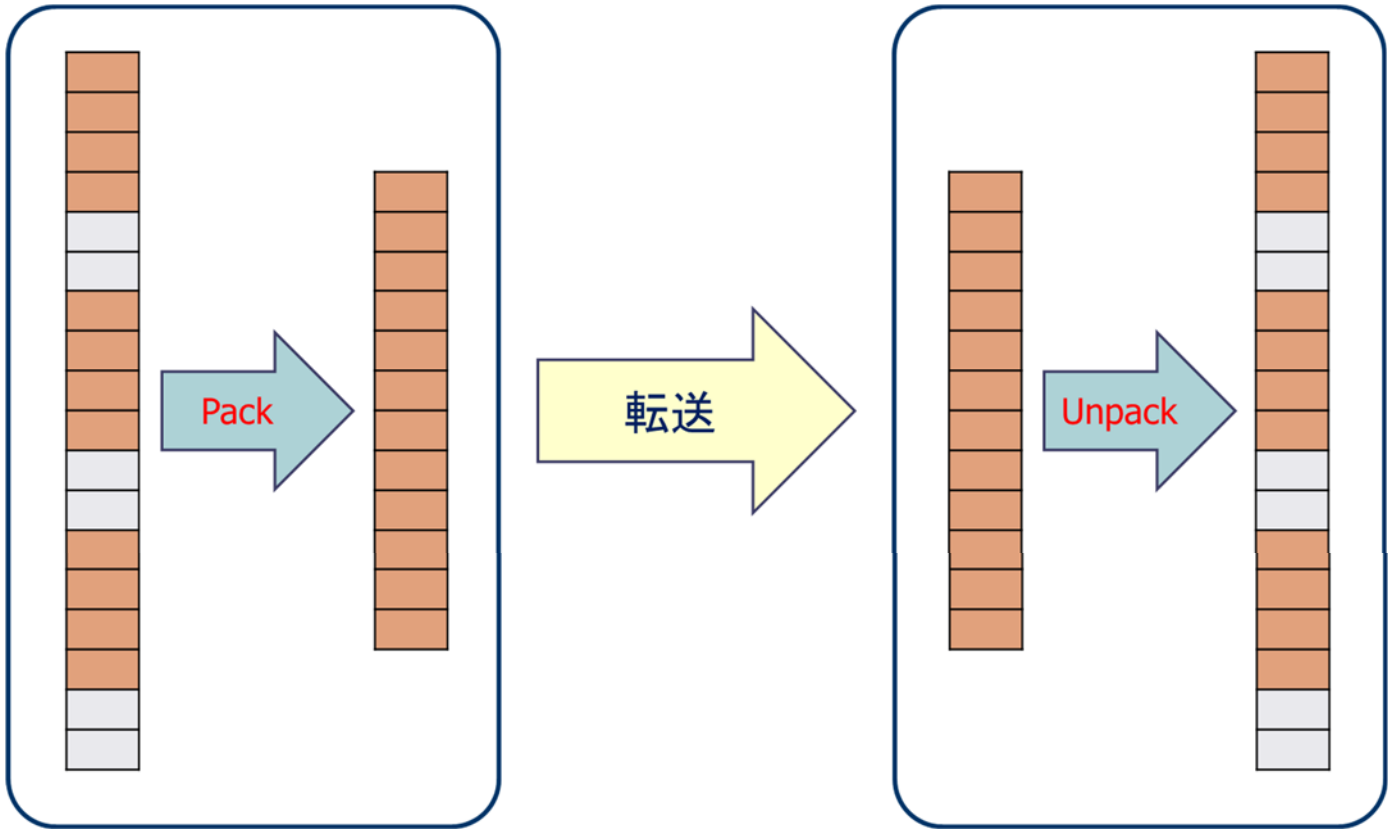
【使用例】

```
MPI_Datatype newtype;
MPI_Type_vector(3, 4, 6, MPI_DOUBLE, &newtype);
MPI_Type_commit( &newtype );

if( myrank == source ) {
   MPI_Send( buf, 1, newtype, dest, tag, COMM );
} else {  // rank番号 dest で受け取る場合
   MPI_Recv( buf, 1, newtype, source, tag, COMM, status);
}
```

派生データ型データの転送における内部処理

【送信側】
不連続なデータを連続なデータに収集（Pack処理）した後に送信



【受信側】
データを受信した後に，元の不連続な順番にデータを展開（Unpack処理）

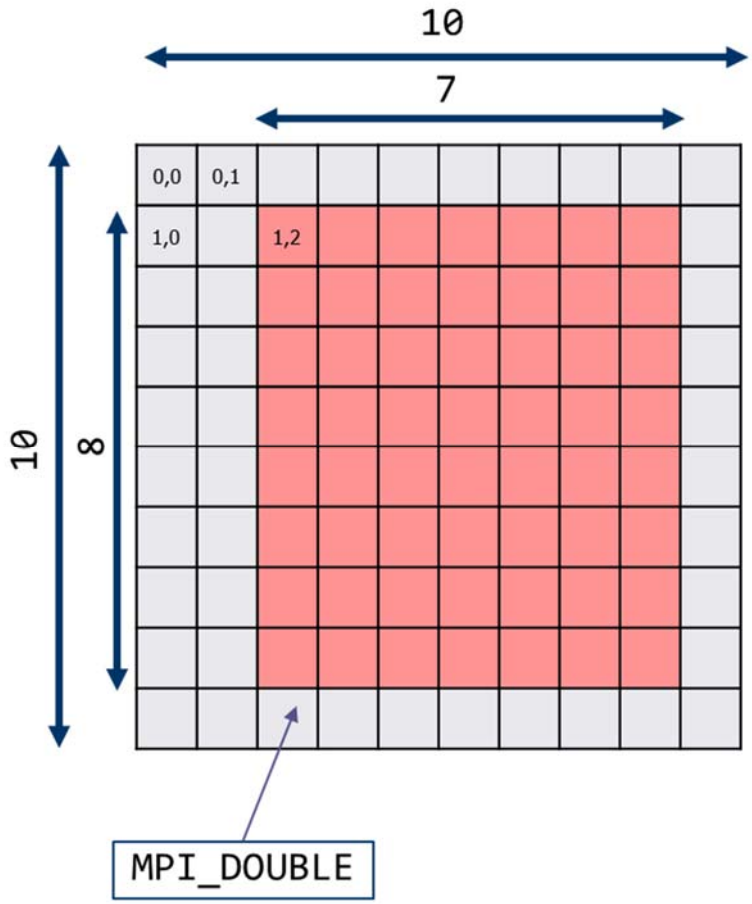
派生データ型の生成関数：MPI_Type_create_subarray

```
【C】int MPI_Type_create_subarray( int ndims, int array_of_sizes[], int array_of_subsizes[],
                                   int array_of_starts[], int order, MPI_Datatype otype,
                                   MPI_Datatype *ntype );

【F】mpi_type_create_subarray( ndims, array_of_sizes, array_of_subsizes, array_of_starts,
                               order, otype, ntype, ierr )
```

- ndims: ベースになる配列の次元数
- array_of_sizes: ベースになる配列の各次元の大きさ（サイズはndims）
- array_of_subsizes: 部分配列の大きさ（サイズはndims）
- array_of starts: 部分配列の開始地点（サイズはndims）
  【注意】Fortranでも 0から始まる.
- order: MPI_ORDER_C，またはMPI_ORDER_FORTRAN
- otype: 元々のデータタイプ
- ntype: 新しいデータタイプ名
- ierr: 戻りコード（整数型）

【例】 2次元データの内部（赤い部分）をアクセスする派生データ型の作成




```
int ndims;
int o_sizes[2];
int n_sizes[2];
int starts[2];
MPI_Datatype ntype;

ndims = 2
o_sizes[0] = 10;
o_sizes[1] = 10;
n_sizes[0] = 7;
n_sizes[1] = 8;
starts[0]  = 2;
starts[1]  = 1;

MPI_Type_create_subarray(ndims, o_sizes, n_sizes,
starts, MPI_ORDER_C, MPI_DOUBLE, &ntype);
```

演習問題

- 16プロセスのMPIプログラムにおいて，起動時のコミュニケータ MPI_COMM_WORLDを，2つのコミュニケータに分割せよ．また，
  4つのコミュニケータに分割せよ．
  - 複数のコミュニケータが出来ていることを，各プロセスの新しいランク番号を出力して確認せよ．
  - また，同時放送（MPI_Bcast）が新コミュニケータ内に閉じていることを確認せよ．
    - それぞれの新コミュニケータのランク0 から他のランクに同時放送する．

【例】8 MPIプロセスのプログラムを
2つのコミュニケータに分割