KOBE HPC スプリングスクール 2021（中級）

MPI（片方向通信）

2021年3月8日

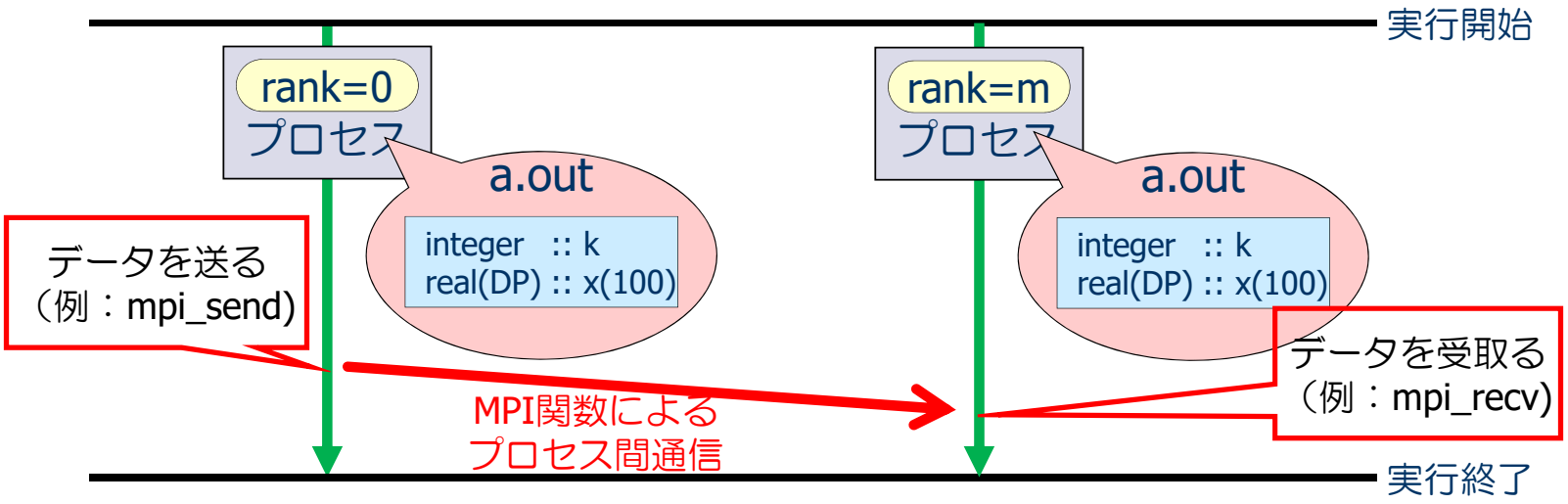
神戸大学大学院システム情報学研究科計算科学専攻
横川三津夫

講義の内容

- 双方向通信
- 片方向通信
- 演習問題

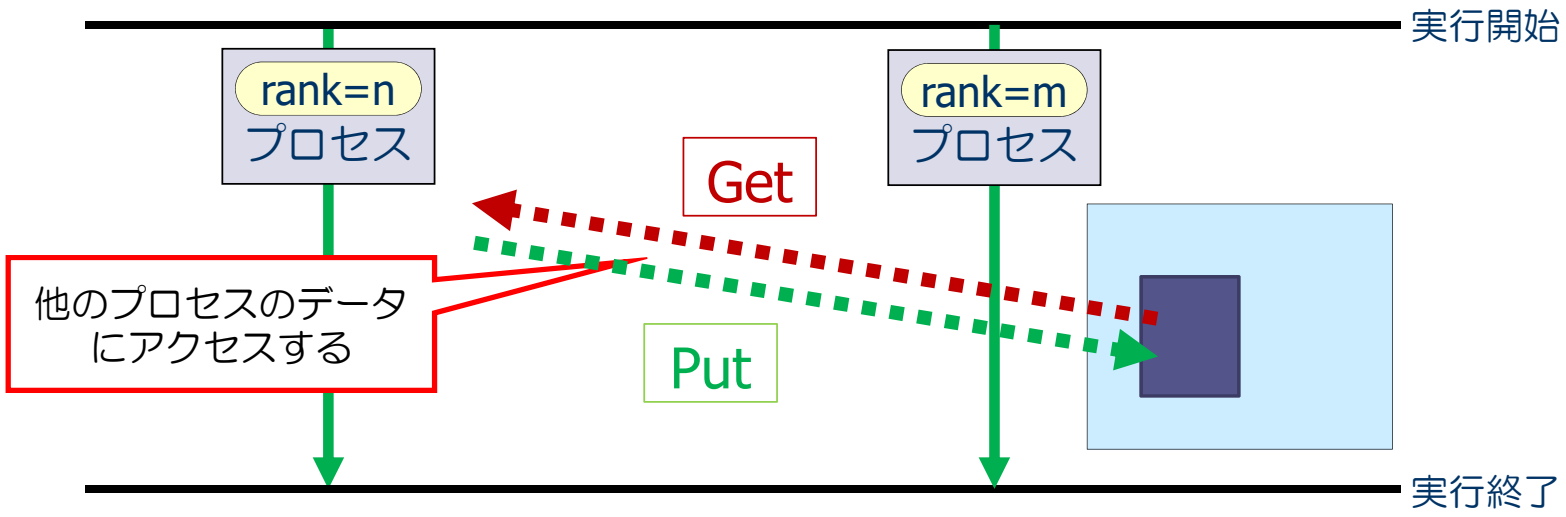
双方向通信（One-to-one communication）

- 送信側と受信側で，対応する関数を呼ぶ．
- MPI_Send，MPI_Recv などの対の組合せ



片方向通信（One-sided communication）

- 通信相手の状態に関係なく，通信相手のプロセスのデータをアクセスする通信方法
  - Get　　相手のプロセスのデータを獲得
  - Put　　相手のプロセスにデータを挿入



片方向通信の利点

- プロセス間の同期待ちを削減
  - 性能向上の可能性
- データコピーの回数を削減
  - 双方向通信では，MPIライブラリ内の中間バッファを用いた実装になっている.
- RDMA（Remote Direct Memory Access）機構を持つ計算機システムでは，高速実行が可能となる.
  - 計算とデータ通信のオーバーラップが可能
- 大きなデータ通信において高速実行される場合がある.

Windowオブジェクトの生成

```
【C】int MPI_Win_create(void *base, MPI_Aint size, int disp_unit, MPI_Info info,
                        MPI_Comm comm, MPI_Win *win )

【F】mpi_win_create( base, size, disp_unit, info, comm, win, ierr )
```

RMA操作に必要なWindowオブジェクトを生成する.

[Input]

- base: windowの先頭アドレス
- size: windowのサイズ（MPI_Aint型，バイト単位で指定）
  - Fortranの場合，integer(kind=MPI_ADDRESS_KIND):: size とする.
- disp_unit: ずれ当りのサイズ（整数型，バイト単位で指定）
  - MPI_INTEGER，MPI_REALなどではない．MPI_Get，MPI_Putなどで使用する.
- info: 情報（Fortranでは整数型，CではMPI_Info型）
  - infoは今回は使わないので，MPI_INFO_NULL とすればよい.
  - infoを使う場合には，MPI_Info_create 関数でinfo オブジェクトを作成する必要がある.
- comm: コミュニケータ（MPI_COMM_WORLDなど）

[Output]

- win: Windowオブジェクト（Fortranでは整数型，CではMPI_Win型）
  - MPI_get，MPI_Putなどで利用
- ierr: 戻りコード（整数型）

Windowオブジェクトの開放

```
【C】int MPI_Win_free( MPI_Win *win )

【F】mpi_win_free( win, ierr )
```

生成したWindowオブジェクトを開放する.

[Input/Output]

- win:　　生成したWindowオブジェクト

[Output]

- ierr:　　戻りコード（整数型）

リモートプロセスのデータ取得

```
【C】int MPI_Get( void *oaddr, int ocount, MPI_Datatype odatatype,
                  int target_rank, MPI_Aint tdisp, int tcount, MPI_Datatype tdatatype, MPI_Win win)

【F】mpi_get( oaddr, ocount, odatetype, target_rank, tdisp, tcount, tdatatype, win, ierr )
```

[Input/Output]

- oaddr: 自分のプロセスのデータを格納する変数の先頭アドレス
- ocount: データの個数（整数型）
- odatatype: データの型，MPI_INTEGER，MPI_REALなど．CではMPI_INTなど．
- target_rank: リモートプロセスのMPIランク番号
- tdisp: 先頭からのずれ（MPI_Aint型）
  - 獲得する変数の先頭アドレスは，base ＋ tdisp×disp_unit
  - Fortranの場合，integer(kind=MPI_ADDRESS_KIND):: tdisp と宣言する．
- tcount: ターゲット側のデータの個数（整数型）
- tdatatype: ターゲット側のデータの型
  - FortranではMPI_INTEGER，MPI_REALなど．CではMPI_INTなど
- win: 通信するwindowオブジェクト

[Output]

- ierr: 戻りコード（整数型）

リモートプロセスへのデータ書き込み

```
【C】int MPI_Put( const void *oaddr, int ocount, MPI_Datatype odatatype,
                  int target_rank, MPI_Aint tdisp, int tcount, MPI_Datatype tdatatype, MPI_Win win)

【F】mpi_put( oaddr, ocount, odatetype, target_rank, tdisp, tcount, tdatatype, win, ierr )
```

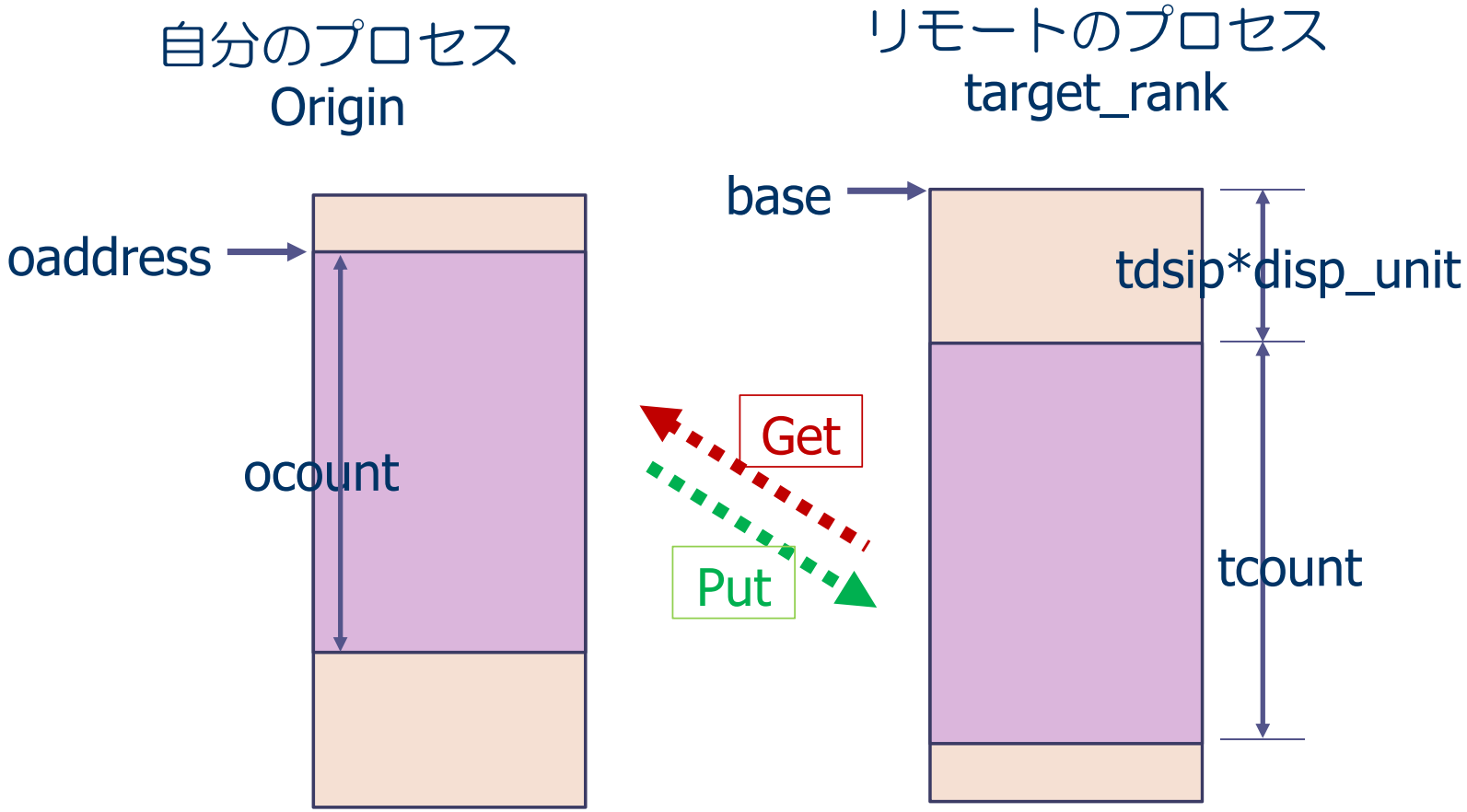
[Input/Output]

- oaddr: 自分のプロセスのデータを格納する変数の先頭アドレス
- ocount: データの個数（整数型）
- odatatype: データの型，MPI_INTEGER，MPI_REALなど．CではMPI_INTなど．
- target_rank: リモートプロセスのMPIランク番号
- tdisp: 先頭からのずれ（MPI_Aint）
  - 獲得する変数の先頭アドレスは，base ＋ tdisp×disp_unit
  - Fortranの場合，integer(kind=MPI_ADDRESS_KIND):: tdisp と宣言する．
- tcount: ターゲット側のデータの個数（整数型）
- tdatatype: ターゲット側のデータの型
  - FortranではMPI_INTEGER，MPI_REALなど．CではMPI_INTなど
- win: 通信するwindowオブジェクト

[Output]

- ierr: 戻りコード（整数型）

片方向通信関数の引数の意味



片方向通信の同期

```
【C】int MPI_Win_fence( int assert, MPI_Win win )

【F】mpi_win_fence( assert, win, ierr )
```

winオブジェクトの片方向通信関数の同期を取る．fence関数のところで片方向通信が終了する．

[Input/Output]

- assert:　Windowの状態の確認用．
  通常は 0でよい．MPI_MODE_NOPRECEDEなど
- win:　生成したWindowオブジェクト

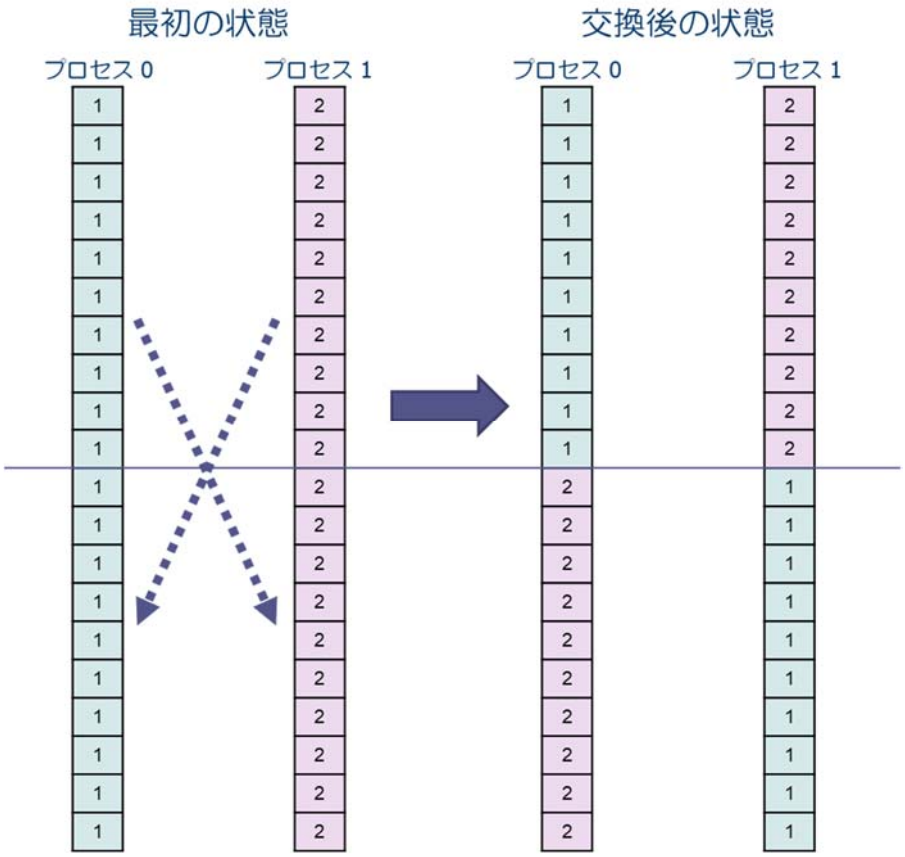
[Output]

- ierr:　戻りコード（整数型）

演習問題

- 2つのMPIプロセスにおいて，以下のプログラムを作れ・
  1. 長さ2nの配列を用意する．配列変数の型は，どんな型（整数型，実数型）でも良い．
  2. 最初の状態としてプロセス0 では配列に1を代入し，プロセス1では配列に2を代入する．
  3. プロセス0の配列の前半部分（長さ n）をプロセス1の配列の後半（長さ n）にコピーし，プロセス１の配列の前半部分（長さ n）をプロセス0の配列の後半（長さ n）にコピーする．

- この動作について，Send – Recv の組合せと，Put – Get の組合せの2つのプログラムを作り，結果を確認すること．
  - n=10 でやると，配列を出力することにより動作を確認できる．



プログラム・スケルトン

send-recv

```
# ヘッダ
# 配列を準備

# mpiの初期化



if( myrank == 0 ) then
   rank = 0 の処理（sendして，recvする）
else
   rank = 1 の処理（recvして，sendする）
endif


# 結果の確認
# MPIの終了
```

put-get

```
# ヘッダ
# 配列を準備

# mpiの初期化

# put，get用のWindowをセット
                                    // プロセス0側だけから操作
if( myrank == 0 ) then
    プロセス0の配列の前半部分を，プロセス1の配列の後半部分に書き込む（Put）

    プロセス1の配列の前半部分を，プロセス0の配列の後半部分にもらう（Get）
endif
# put，getの同期待ち．MPI_Win_fence

# 結果の確認
# MPIの終了
```

参考：send関数, receive関数

【復習】 1対1通信 – 送信関数（送り出し側）【C】

```
int MPI_Send(void *buff, int count, MPI_Datatype datatype, int dest, int tag, MPI_Comm comm)
```

※ ランク番号destのプロセスに，変数buffの値を送信する.

- buff: 送信するデータの変数名（先頭アドレス）
- count: 送信するデータの数（非負の整数型）
- datatype: 送信するデータの型（handle）
  - MPI_CHAR, MPI_INT, MPI_DOUBLEなど
- dest: 送信先プロセスのランク番号（整数型）
- tag: メッセージ識別番号．送るデータを区別するための番号（整数型）
- comm: コミュニケータ（例えば，MPI_COMM_WORLD）

※ 関数の戻りコードは，エラーコード

【復習】 1対1通信 – 受信関数（受け取り側）【C】

```
int MPI_Recv(void *buff, int count, MPI_Datatype datatype, int source, int tag, MPI_Comm comm,
             MPI_Status *status)
```

※ ランク番号sourceのプロセスから送られたデータを，変数buffに格納する．

- buff:　　受信するデータのための変数名（先頭アドレス）
- count:　　受信するデータの数（非負の整数型）
- datatype:　受信するデータの型（handle）
  - MPI_CHAR，MPI_INT，MPI_DOUBLEなど
- source:　　送信してくる相手プロセスのランク番号（整数型）
- tag:　　メッセージ識別番号．送られて来たデータを区別するための番号（整数型）
- comm:　　コミュニケータ（例えば，MPI_COMM_WORLD）
- status:　　通信の状態を格納するオブジェクト．MPI_Sendにはないので注意．

※関数の戻りコードは，エラーコード

【復習】 1対1通信 – 送信関数（送り出し側）【Fortran】

```
call mpi_send( buff, count, datatype, dest, tag, comm, ierr )
```

※ ランク番号destのプロセスに，変数buffの値を送信する.

- buff: 送信するデータの変数名（先頭アドレス）
- count: 送信するデータの数（整数型）
- datatype: 送信するデータの型（整数型）
  - MPI_INTEGER, MPI_DOUBLE_PRECISION（MPI_REAL, MPI_CHARACTER など
- dest: 送信先プロセスのランク番号
- tag: メッセージ識別番号．送るデータを区別するための番号
- comm: コミュニケータ（例えば，MPI_COMM_WORLD）
- ierr: 戻りコード（整数型）

【復習】 1対1通信 – 受信関数（受け取り側）【Fortran】

```
call mpi_recv( buff, count, datatype, source, tag, comm, status, ierr )
```

※ ランク番号sourceのプロセスから送られたデータを，変数buffに格納する．

- buff: 受信するデータのための変数名（先頭アドレス）
- count: 受信するデータの数（非負の整数型）
- datatype: 受信するデータの型
  - MPI_INTEGER, MPI_REAL, MPI_DOUBLE_PRECISION, MPI_CHARACTER など
- source: 送信してくる相手プロセスのランク番号（整数型）
- tag: メッセージ識別番号．送られて来たデータを区別するための番号（整数型）
- comm: コミュニケータ（例えば，MPI_COMM_WORLD）
- status: 通信の状態を格納するオブジェクト．MPI_Sendにはないので注意．
- ierr: 戻りコード（整数型）