行列-行列積（2）

東京大学情報基盤センター　准教授　塙　敏博

2019年6月25日（火）10:25-12:10

レポートおよびコンテスト課題
（締切：
2019年8月5日（月）24時　厳守

講義日程（工学部共通科目）

1. ~~4月9日：ガイダンス~~
2. ~~4月16日~~
   - ~~並列数値処理の基本演算（座学）~~
3. ~~4月23日：スパコン利用開始~~
   - ~~ログイン作業、テストプログラム実行~~
4. ~~5月7日~~
   - ~~高性能プログラミング技法の基礎1（階層メモリ、ループアンローリング）~~
5. ~~5月21日~~
   - ~~高性能プログラミング技法の基礎2（キャッシュブロック化）~~
6. ~~5月28日~~
   - ~~行列-ベクトル積の並列化~~
7. ~~6月4日~~
   - ~~べき乗法の並列化~~
8. ~~6月11日~~
   - ~~行列-行列積の並列化（1）~~
9. 6月25日
   - 行列ー行列積の並列化（2）
10. 7月2日
    - LU分解法（1）
    - コンテスト課題発表
11. 7月9日
    - LU分解法（2）
12. 7月16日
    - LU分解法（3）、非同期通信
13. **7月17日**
    - RB-Hお試し、研究紹介他

53nd TOP500 List (June, 2019)

R_{max}: Performance of Linpack (TFLOPS)
R_{peak}: Peak Performance (TFLOPS),
Power: kW

http://www.top500.org/

	Site	Computer/Year Vendor	Cores	R_{max} (TFLOPS)	R_{peak} (TFLOPS)	Power (kW)
1	**Summit, 2018, USA** DOE/SC/Oak Ridge National Laboratory	IBM Power System AC922, IBM POWER9 22C 3.07GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband	2,414,592	148,600 (= 148.6 PF)	200,795	**10,096**
2	**Sieera, 2018, USA** DOE/NNSA/LLNL	IBM Power System S922LC, IBM POWER9 22C 3.1GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband	1,572,480	94,640	125,712	**7,438**
3	**Sunway TaihuLight, 2016, China** National Supercomputing Center in Wuxi	Sunway MPP, Sunway SW26010 260C 1.45GHz, Sunway	10,649,600	93,015	125,436	**15,371**
4	**Tianhe-2A, 2018, China** National Super Computer Center in Guangzhou	TH-IVB-FEP Cluster, Intel Xeon E5-2692v2 12C 2.2GHz, TH Express-2, Matrix-2000	4,981,760	61,445	100,679	**18,482**
5	**Frontera, 2019, USA** Texas Advanced Computing Center	Dell C6420, Xeon Platinum 8280 28c 2.7GHz, Mellanox Infiniband HDR	448,448	23,516	38,746	
6	**Piz Daint, 2017, Switzerland** Swiss National Supercomputing Centre (CSCS)	Cray XC50, Xeon E5-2690v3 12C 2.6GHz, Aries interconnect , NVIDIA Tesla P100	387,872	21,230	27,154	**2,384**
7	**Trinity, 2017, USA** DOE/NNSA/LANL/SNL	Cray XC40, Intel Xeon Phi 7250 68C 1.4GHz, Aries interconnect	979,072	20,159	41,461	**7,578**
8	**ABCI (AI Bridging Cloud Infrastructure), 2018, Japan** National Institute of Advanced Industrial Science and Technology (AIST)	PRIMERGY CX2550 M4, Xeon Gold 6148 20C 2.4GHz, NVIDIA Tesla V100 SXM2, Infiniband EDR	391,680	19,880	32,577	**1,649**
9	**SuperMUC-NG, 2018, Germany** Leibniz Rechenzentrum	Lenovo, ThinkSystem SD650, Xeon Platinum 8174 24C 3.1GHz, Intel Omni-Path	305,856	19,477	26,874	
10	**Lassen, 2019, USA** DOE/NNSA/LLNL	IBM Power System S922LC, IBM POWER9 22C 3.1GHz, NVIDIA Volta V100, Dual-rail Mellanox EDR Infiniband	288,288	18,200	23,047	
16	**Oakforest-PACS, 2016, Japan** Joint Center for Advanced High Performance Computing	PRIMERGY CX1640 M1, Intel Xeon Phi 7250 68C 1.4GHz, Intel Omni-Path	556,104	13,556	24,913	2,719

HPCG Ranking (June, 2019)

	Computer	Cores	HPL Rmax (Pflop/s)	TOP500 Rank	HPCG (Pflop/s)
1	**Summit**	2,414,592	148,600	1	2.926
2	**Sierra**	1,572,480	94.640	2	1.796
3	**K computer**	705,024	10.510	20	0.603
4	**Trinity**	979,072	20,159	7	0.546
5	**ABCI**	391,680	19,880	8	0.509
6	**Piz Daint**	387,872	21.230	6	0.497
7	**Sunway TaihuLight**	10,649,600	93.015	3	0.481
8	**Nurion (KISTI, Korea)**	570,020	13.929	15	0.391
9	**Oakforest-PACS**	556,104	13.555	16	0.385
10	**Cori (NERSC/LBNL, USA)**	632,400	14.015	14	0.355

http://www.hpcg-benchmark.org/

Green 500 Ranking (June, 2019)

http://www.top500.org/

	TOP 500 Rank	System	Cores	HPL Rmax (Pflop/s)	Power (MW)	GFLOPS/W
1	472	Shoubu system B, Japan	953,280	1,063	60	17.604
2	470	DGX SaturnV Volta, USA	22,440	1,070	97	15.113
3	1	Summit, USA	2,414,592	148,600	10,096	14.719
4	8	ABCI, Japan	391,680	19,880	1,649	14.423
5	394	MareNostrum P9 CTE, Spain	18,360	1,145	81	14.131
6	25	TSUBAME 3.0, Japan	135,828	8,125	792	13.704
7	444	PANGEA III, France	291,024	17,860	1,367	13.065
8	2	Sierra, USA	1,572,480	94,640	7,438	12.723
9	43	Advanced Computing System (PreE), China	163,840	4,325	380	11.382
10	23	Taiwania 2, Taiwan	170,352	900	798	11.285
13	June’18	Reedbush-L, U.Tokyo, Japan	16,640	806	79	10.167
19		Reedbush-H, U.Tokyo, Japan	17,760	802	94	8.576

IO 500 Ranking (June, 2019)

http://www.io500.org/

	Site	Computer	File system	Client nodes/ procs	IO500 Score	BW (GiB/s)	MD (kIOP/s)
1	University of Cambridge, UK	Data Accelerator	Dell EMC Lustre	512 8192	**620.69**	162.05	2377.44
2	Oak Ridge National Laboratory, USA	Summit	IBM Spectrum Scale	504 1008	**330.56**	88.20	1238.93
3	JCAHPC, Japan	Oakforest-PACS	DDN IME	2048 2048	**275.65**	492.06	154.41
4	KISTI, Korea	NURION	DDN IME	2048 4096	**156.91**	554.23	44.43
5	CSIRO, Australia	bracewell	Dell/ThinkParQ BeeGFS	26 260	**140.58**	69.29	285.21
6	DDN	IME140	DDN IME	17 272	**112.67**	90.34	140.52
7	DDN Colorado	DDN IME140	DDN IME	10 160	**109.42**	75.79	157.96
8	DDN	AI400	DDN Lustre	10 160	**104.34**	19.65	553.98
9	KAUST, Saudi	Shaheen2	Cray DataWarp	1024 8192	**77.37**	496.81	12.05
10	University of Cambridge, UK	Data Accelerator	Dell EMC BeeGFS	184 5888	**74.58**	58.81	94.57

講義の流れ

1. 行列-行列積（2）のサンプルプログラムの実行
2. サンプルプログラムの説明
3. 演習課題（2）：ちょっと難しい完全分散版
4. 並列化のヒント

行列-行列積の演習の流れ

- 演習課題（1）
  - 簡単なもの（30分程度で並列化）
  - 通信関数が一切不要
- 演習課題（2）
  - ちょっと難しい（1時間以上で並列化）
  - 1対1通信関数が必要

サンプルプログラムの実行（行列-行列積（2））

行列-行列積のサンプルプログラムの注意点

- C言語版/Fortran言語版の共通ファイル名
  Mat-Mat-d-ofp.tar.gz
- ジョブスクリプトファイルmat-mat-d.bash　中のキュー名を
  lecture-flat から
  lecture5-flat　(工学部共通科目)、
  に変更し、
  pjsub　してください。
  - lecture-flat : 実習時間外のキュー
  - lecture5-flat :実習時間内のキュー
  - グループ名 : gt25

行列-行列積(2)のサンプルプログラムの実行

- 以下のコマンドを実行する
  $ cd /work/gt25/t25XXX
  $ cp /work/gt25/z30105/Mat-Mat-d-ofp.tar.gz ./
  $ tar xvfz Mat-Mat-d-ofp.tar.gz
  $ cd Mat-Mat-d
- 以下のどちらかを実行
  $ cd C ：C言語を使う人
  $ cd F ：Fortran言語を使う人
- 以下共通
  $ make
  $ pjsub mat-mat-d.bash
- 実行が終了したら、以下を実行する
  $ cat mat-mat-d.bash.oXXXXXX

行列-行列積のサンプルプログラムの実行（C言語版）

- 以下のような結果が見えれば成功

```
Error! in ( 0 , 2 )-th argument in PE 0
Error! in ( 0 , 2 )-th argument in PE 61
 …
N  = 2176
Mat-Mat time  = 0.000896 [sec.]
 22999044.714267 [MFLOPS]
…
N  = 2176
Mat-Mat time  = 0.000285 [sec.]
 72326702.959176 [MFLOPS]
…
N  = 2176
Mat-Mat time  = 0.000237 [sec.]
 86952122.772853 [MFLOPS]
```

並列化が完成していないのでエラーが出ます。ですが、これは正しい動作です

行列-行列積のサンプルプログラムの実行（Fortran言語）

- 以下のような結果が見えれば成功

```
Error! in ( 1 , 3 )-th argument in PE 0
Error! in ( 1 , 3 )-th argument in PE 1033
...
 NN  =        2176
 Mat-Mat time  =   1.497983932495117E-003
 MFLOPS =    13756232.6624224
...
 NN  =        2176
 Mat-Mat time  =   1.003026962280273E-003
 MFLOPS =    20544428.2904683
...
 NN  =        2176
 Mat-Mat time  =   1.110076904296875E-003
 MFLOPS =    18563232.3492268
...
```

並列化が
完成して
いないので
エラーが出ます。
ですが、
これは正しい
動作です。

サンプルプログラムの説明

- #define N 2176
  - 数字を変更すると、行列サイズが変更できます
- #define DEBUG 1
  - 「0」を「1」にすると、行列-行列積の演算結果が検証できます。
- MyMatMat関数の仕様
  - Double型の行列A（（N/NPROCS）×N行列）と B（（N×（N/NPROCS）行列））の行列積をおこない、 Double型の（N/NPROCS)×N行列Cに、その結果が入ります。

Fortran言語のサンプルプログラムの注意

- 行列サイズ変数が、NNとなっています。
  integer, parameter :: NN=2176

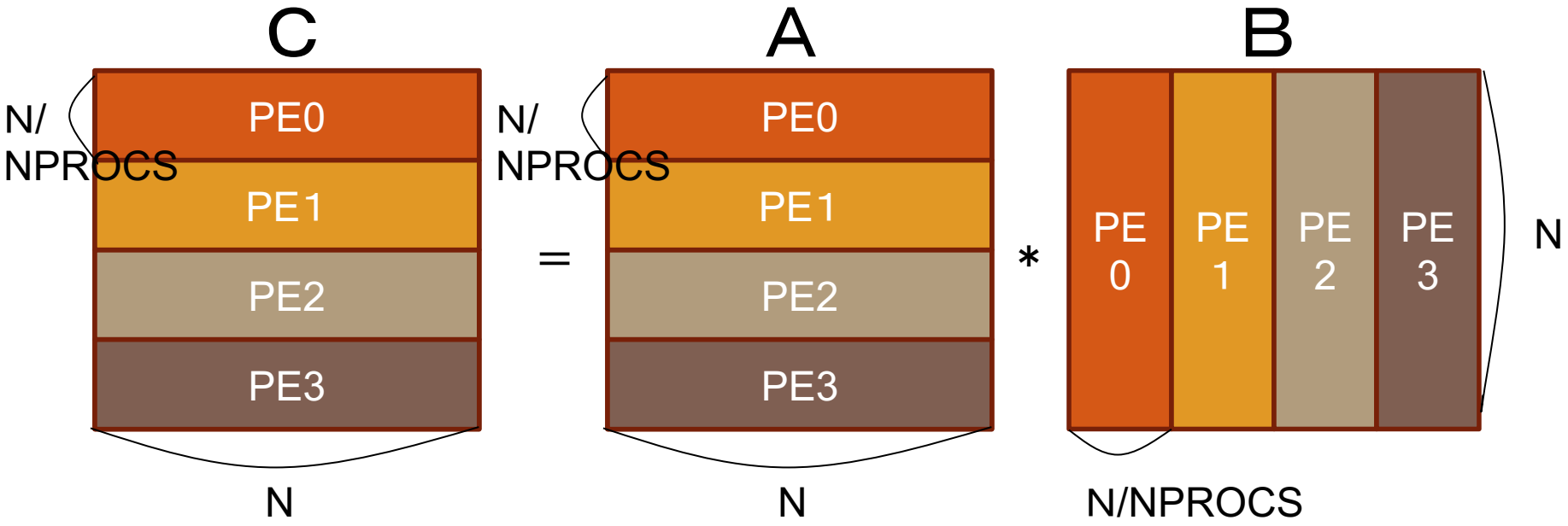
演習課題（1）

- MyMatMat関数（手続き）を並列化してください。
  - デバック時は
    #define N 2176
    としてください。

- 行列A、B、Cの初期配置（データ分散）を、十分に考慮してください。

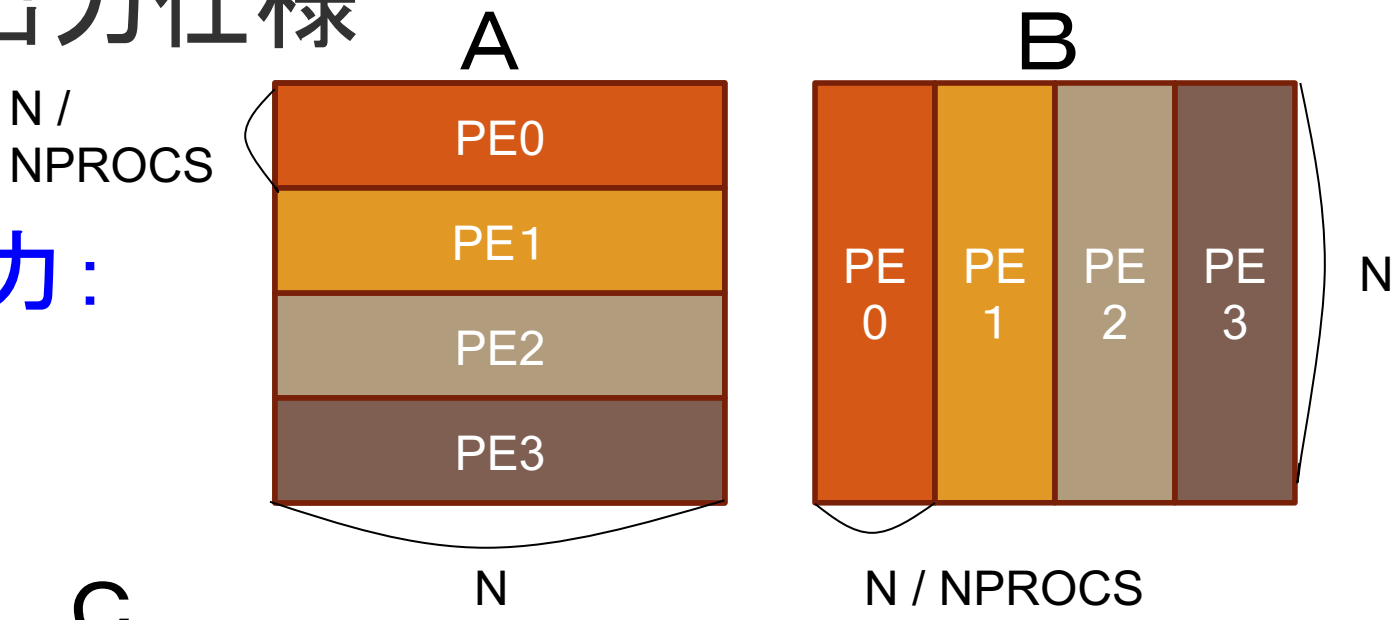
行列A、B、Cの初期配置

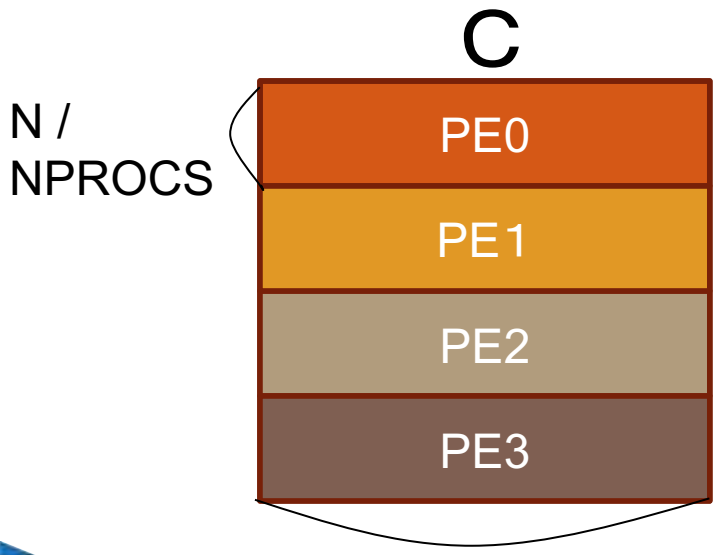
- 行列A、B、Cの配置は以下のようになっています。
  （ただし以下は4PEの場合で、実習環境は1088PEです。）



- 1対1通信関数が必要です。
- 行列A、B、Cの配列のほかに、受信用バッファの配列が必要です。

入力と出力仕様



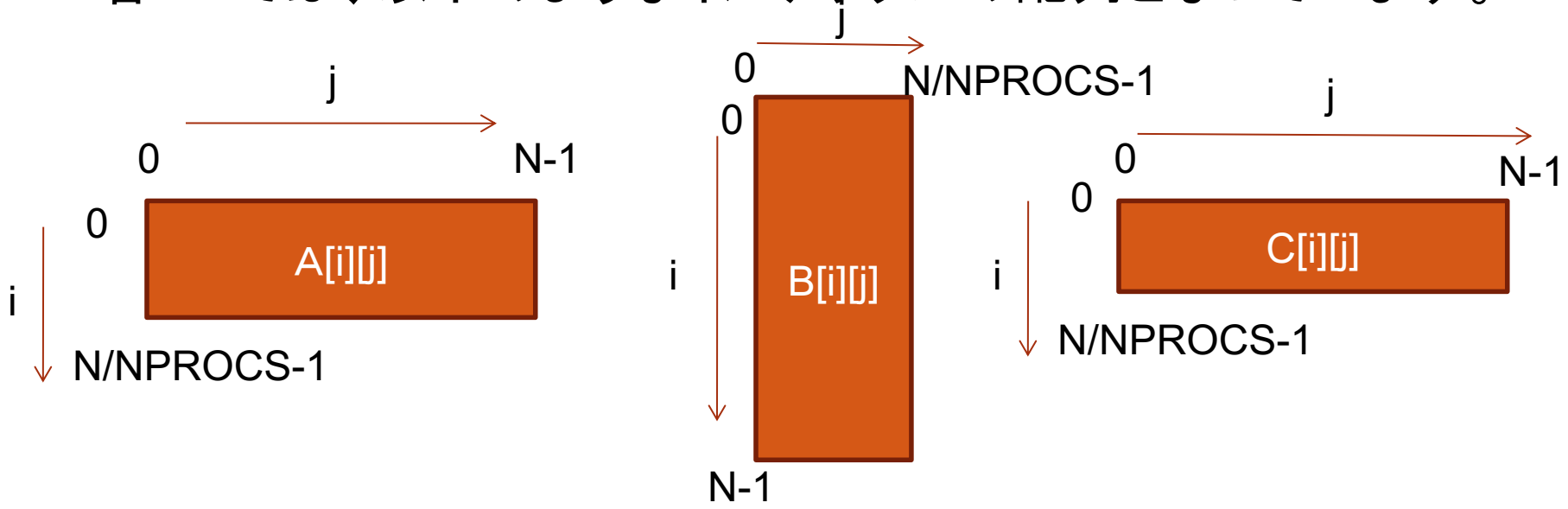


：出力

●この例は4PEの場合ですが、実習環境は1088PEです。

並列化の注意（C言語）

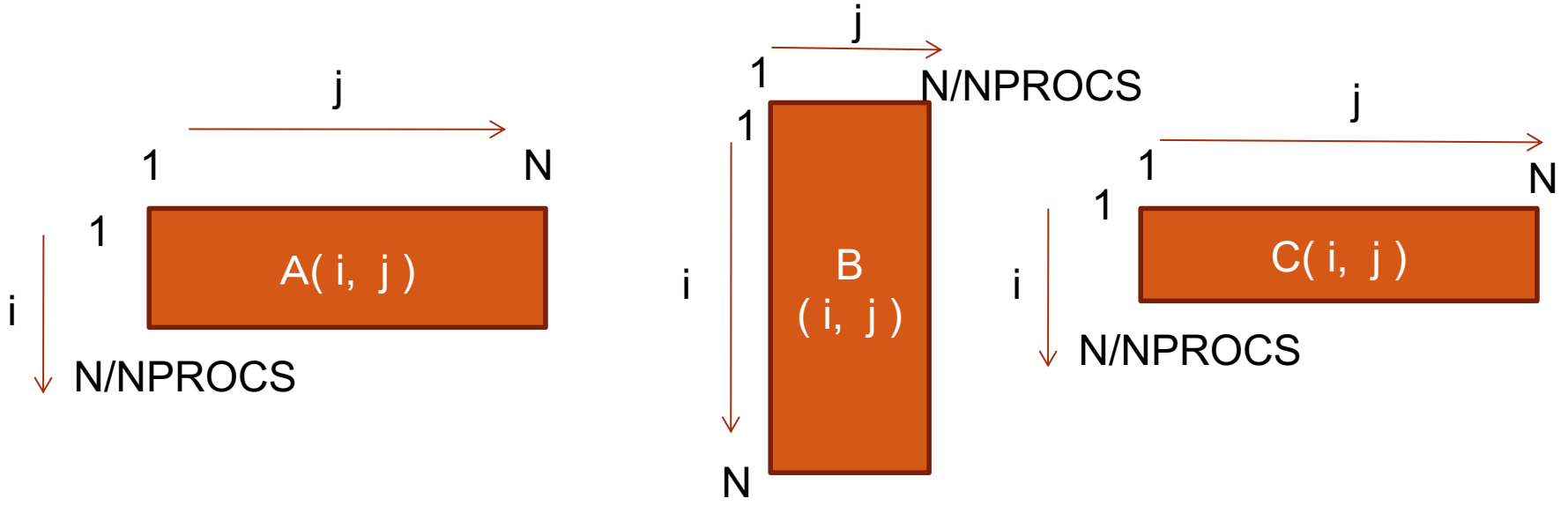
- 各配列は、完全に分散されています。
- 各PEでは、以下のようなインデックスの配列となっています。



- 各PEで行う、ローカルな行列-行列積演算時のインデックス指定に注意してください。

並列化の注意（Fortran言語）

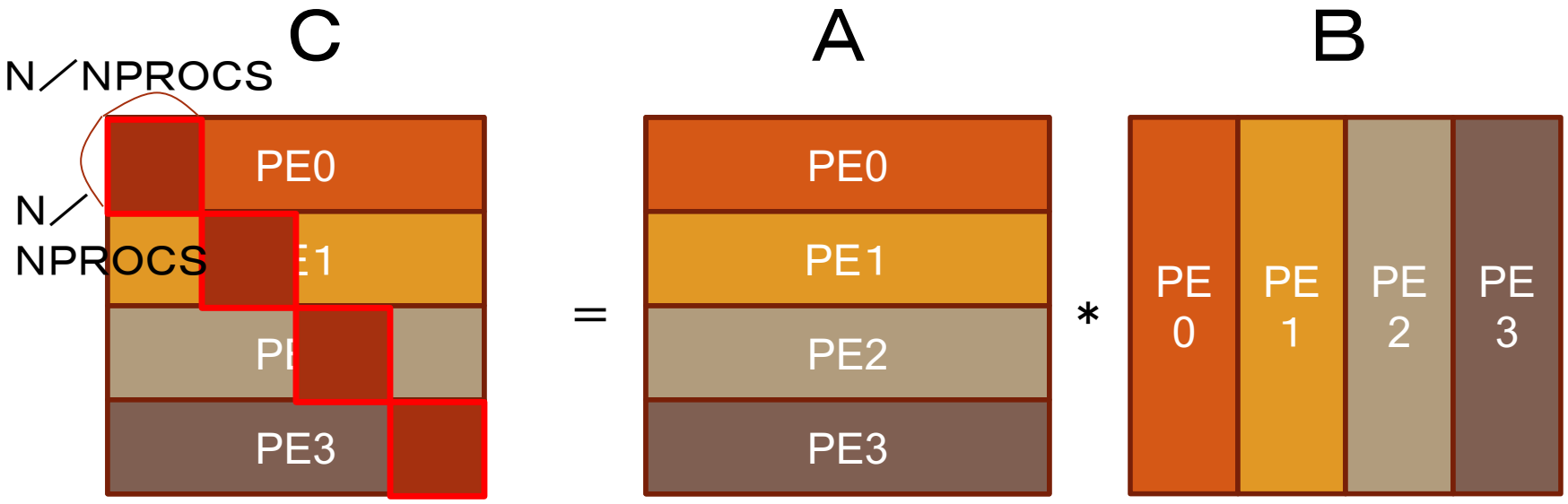
- 各配列は、完全に分散されています。
- 各PEでは、以下のようなインデックスの配列となっています。



- 各PEで行う、ローカルな行列-行列積演算時のインデックス指定に注意してください。

並列化のヒント

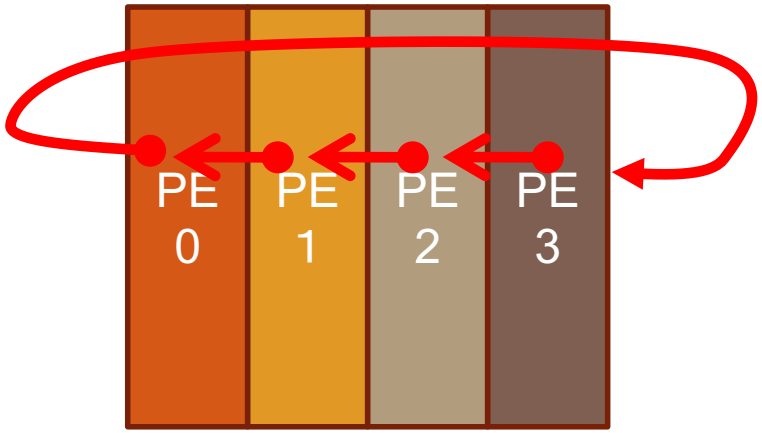
- 行列積を計算するには、各PEで完全な行列Bのデータがないので、行列Bのデータについて通信が必要です。
- たとえば、以下のように計算する方法があります。
- ステップ1



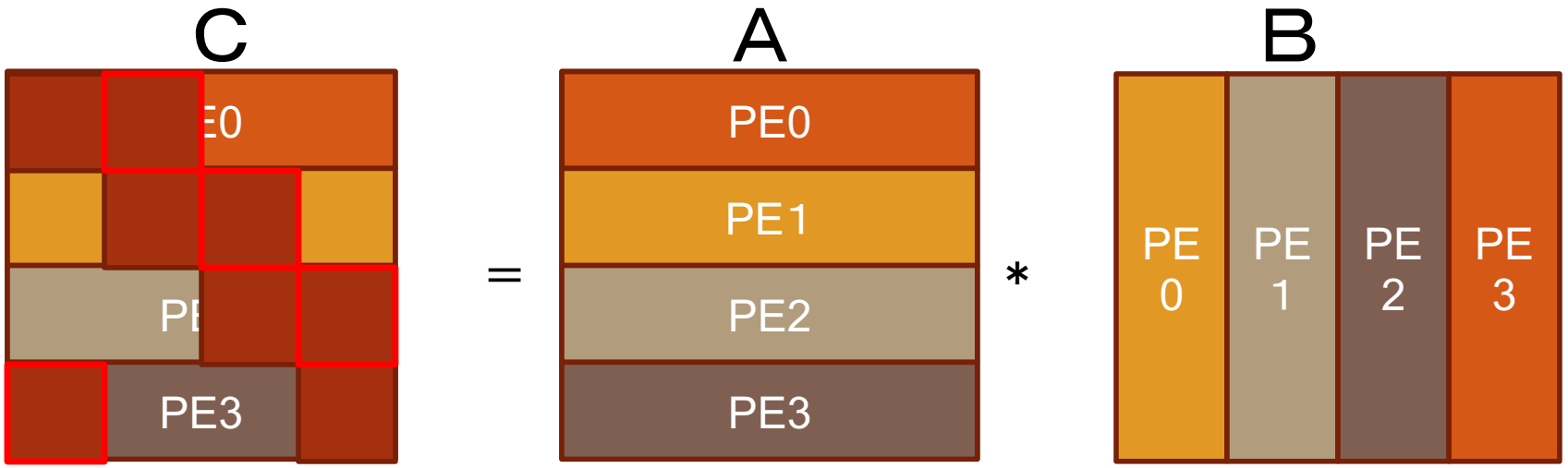
ローカルなデータを使って得られた行列-行列積結果

並列化のヒントＢ

- ステップ２



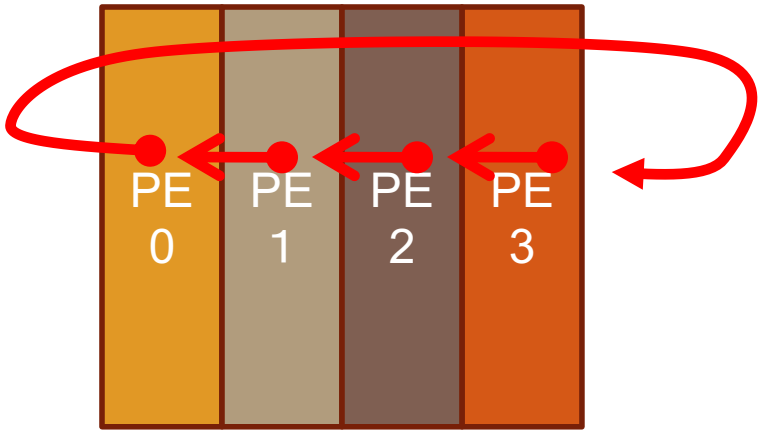
自分の持っているデータを
ひとつ左隣りに転送する
（PE0は、PE3に送る）
【循環左シフト転送】



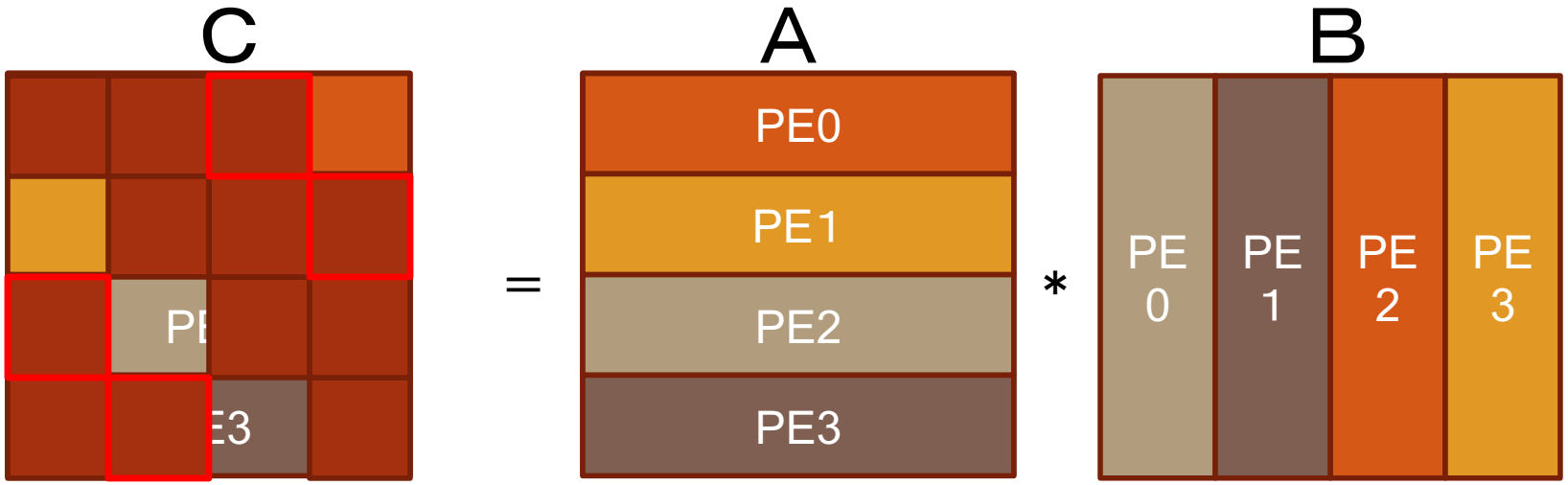
ローカルなデータを使って得られた
行列-行列積結果

並列化のヒントＢ

- ステップ3



自分の持っているデータを
ひとつ左隣りに転送する
（PE0は、PE3に送る）
【循環左シフト転送】



ローカルなデータを使って得られた
行列-行列積結果

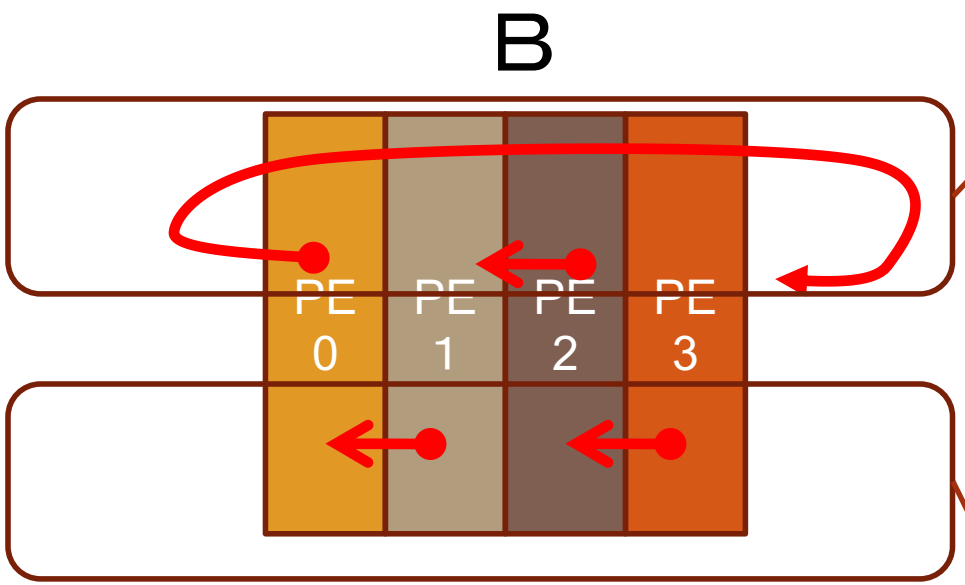
並列化の注意

- 循環左シフト転送を実装する際、全員がMPI_Sendを先に発行すると、その場所で処理が止まる。
  （正確には、動いたり、動かなかったり、する）
  - MPI_Sendの処理中で、場合により、バッファ領域がなくなる。
  - バッファ領域が空くまで待つ（スピンウェイトする）。
  - しかし、バッファ領域不足から、永遠に空かない。
- これを回避するため、以下の実装を行う。
  - PE番号が2で割り切れるPE:
    - MPI_Send();
    - MPI_Recv();
  - それ以外のPE:
    - MPI_Recv();
    - MPI_Send();

それぞれに対応

並列化の注意

- つまり、以下の2ステップで、循環左シフト通信をする



フェーズ1：
2で割り切れるPEがデータを送る

フェーズ2：
2で割り切れないPEがデータを送る

基礎的なMPI関数—MPI_Send

- ierr = MPI_Send(sendbuf, icount, idatatype, idest, itag, icomm);

  - sendbuf : 送信領域の先頭番地を指定する
  - icount : 整数型。送信領域のデータ要素数を指定する
  - idatatype : 整数型。送信領域のデータの型を指定する
  - idest : 整数型。送信したいPEのicomm内でのランクを指定する。
  - itag : 整数型。受信したいメッセージに付けられたタグの値を指定する。
  - icomm : 整数型。プロセッサー集団を認識する番号であるコミュニケータを指定する。
  - ierr (戻り値) : 整数型。エラーコードが入る。

基礎的なMPI関数—MPI_Recv (1/2)

- ierr = MPI_Recv(recvbuf, icount, idatatype, isource, itag, icomm, istatus);

  - recvbuf : 受信領域の先頭番地を指定する。
  - icount : 整数型。受信領域のデータ要素数を指定する。
  - idatatype : 整数型。受信領域のデータの型を指定する。
    - MPI_CHAR (文字型) 、MPI_INT (整数型)、MPI_FLOAT (実数型)、 MPI_DOUBLE(倍精度実数型)
  - isource : 整数型。受信したいメッセージを送信するPEのランクを指定する。
    - 任意のPEから受信したいときは、MPI_ANY_SOURCE を指定する。

基礎的なMPI関数—MPI_Recv (2/2)

- itag： 整数型。受信したいメッセージに付いているタグの値を指定する。
  - 任意のタグ値のメッセージを受信したいときは、MPI_ANY_TAG を指定する。
- icomm： 整数型。PE集団を認識する番号であるコミュニケータを指定する。
  - 通常ではMPI_COMM_WORLD を指定すればよい。
- istatus： MPI_Status型（整数型の配列）。受信状況に関する情報が入る。
  - 要素数がMPI_STATUS_SIZEの整数配列が宣言される。
  - 受信したメッセージの送信元のランクが istatus[MPI_SOURCE]、タグが istatus[MPI_TAG] に代入される。
  - **C言語: MPI_Status istatus;**
  - **Fortran言語: integer istatus(MPI_STATUS_SIZE)**
- ierr(戻り値)： 整数型。エラーコードが入る。

実装上の注意

- タグ(itag)について
  - MPI_Send(), MPI_Recv()で現れるタグ（itag）は、任意のint型の数字を指定してよいです。
  - ただし、同じ値（0など）を指定すると、どの通信に対応するかわからなくなり、誤った通信が行われるかもしれません。
  - 循環左シフト通信では、MPI_Send()と MPI_Recv()の対が、2つでてきます。これらを別のタグにした方が、より安全です。
  - たとえば、一方は最外ループの値iloopとして、もう一方をiloop+NPROCSとすれば、全ループ中でタグがぶつかることがなく、安全です。

さらなる並列化のヒント

以降、本当にわからない人のための資料です。
ほぼ回答が載っています。

並列化のヒント

1. 循環左シフトは、PE総数-1回 必要
2. 行列Bのデータを受け取るため、行列B[][]に関するバッファ行列B_T[][]が必要
3. 受け取ったB_T[][] を、ローカルな行列-行列積で使うため、B[][]へコピーする。
4. ローカルな行列-行列積をする場合の、対角ブロックの初期値：ブロック幅*myid。ループ毎にブロック幅だけ増やしていくが、Nを超えたら0に戻さなくてはいけない。

並列化のヒント（ほぼ回答、C言語）

- 以下のようなコードになる。

```
ib = n/numprocs;
for (iloop=0;  iloop<NPROCS;  iloop++ )  {
    ローカルな行列-行列積  C  =  A * B;
    if (iloop != (numprocs-1) ) {
      if (myid % 2 == 0 ) {
          MPI_Send(B, ib*n, MPI_DOUBLE, isendPE,
                iloop, MPI_COMM_WORLD);
          MPI_Recv(B_T, ib*n, MPI_DOUBLE, irecvPE,
                iloop+numprocs, MPI_COMM_WORLD, &istatus);
      } else {
          MPI_Recv(B_T, ib*n, MPI_DOUBLE, irecvPE,
                iloop, MPI_COMM_WORLD, &istatus);
          MPI_Send(B, ib*n, MPI_DOUBLE, isendPE,
                iloop+numprocs, MPI_COMM_WORLD);
      }
      B[ ][ ]  へ  B_T[ ][ ]  をコピーする;
    }
  }
```

並列化のヒント（ほぼ回答、C言語）

- ローカルな行列-行列積は、以下のようなコードになる。

```
jstart=ib*( (myid+iloop)%NPROCS );
for (i=0;  i<ib;  i++) {
   for(j=0;  j<ib;  j++) {
      for(k=0;  k<n;  k++) {
         C[ i ][ jstart + j ] += A[ i ][ k ] * B[ k ][ j ];
      }
   }
}
```

並列化のヒント（ほぼ回答，Fortran言語）

- 以下のようなコードになる。

```
ib = n/numprocs
do iloop=0,  NPROCS-1
    ローカルな行列-行列積  C  =  A * B
    if (iloop .ne. (numprocs-1) ) then
      if (mod(myid, 2)  .eq. 0 ) then
         call MPI_SEND(B, ib*n, MPI_DOUBLE_PRECISION,  isendPE,
&                iloop, MPI_COMM_WORLD,  ierr)
         call MPI_RECV(B_T,  ib*n,  MPI_DOUBLE_PRECISION,  irecvPE,
&                iloop+numprocs, MPI_COMM_WORLD,  istatus,  ierr)
      else
         call MPI_RECV(B_T,  ib*n,  MPI_DOUBLE_PRECISION,  irecvPE,
&                iloop,  MPI_COMM_WORLD,  istatus,  ierr)
         call MPI_SEND(B,  ib*n,  MPI_DOUBLE_PRECISION,  isendPE,
&                iloop+numprocs,  MPI_COMM_WORLD,  ierr)
      endif
      B  へ  B_T  をコピーする
    endif
  enddo
```

並列化のヒント（ほぼ回答，Fortran言語）

- ローカルな行列-行列積は、以下のようなコードになる。

```
imod = mod( (myid+iloop), NPROCS )
jstart = ib* imod
do i=1, ib
   do j=1, ib
      do k=1, n
        C( i , jstart + j ) = C( i , jstart + j ) +  A( i , k ) * B( k , j )
      enddo
    enddo
 enddo
```

来週へつづく

LU分解法（1）