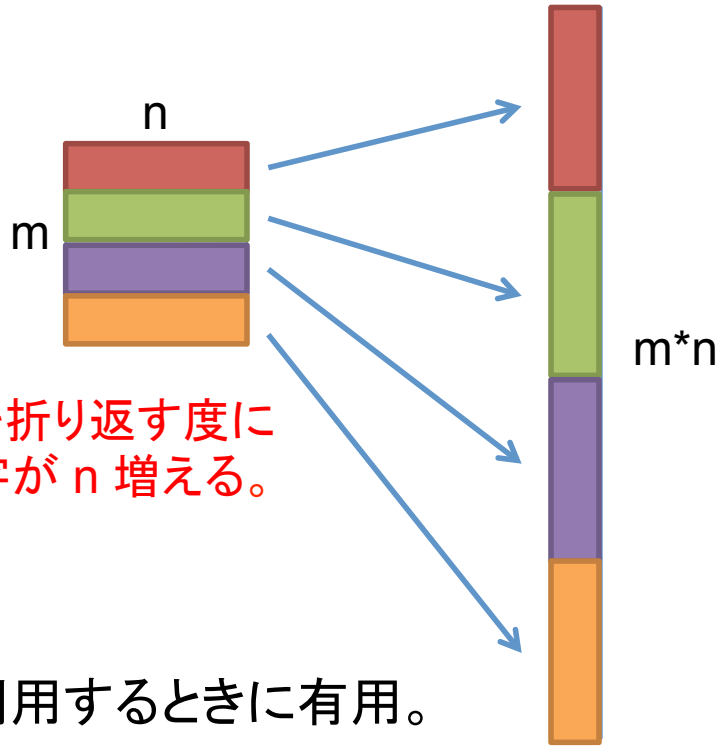
# ハイパフォーマンスコンピューティング(7)

# 2次元配列から1次元配列への変換

- 行列を表現するとき、2次元配列を用いるが、これを1次元配列で表現することができる。
- Aが m x n 行列のとき

  C言語: A[i][j] ⇔ A[i*n+j]

  Fortran: A(j,i) ⇔ A((i-1)*n+j)




- メリット
  - 配列の確保に無駄がなくなる。
  - 関数へポインタを渡すのが簡単になる。
  - Fortranなどの多言語との互換性が増す。

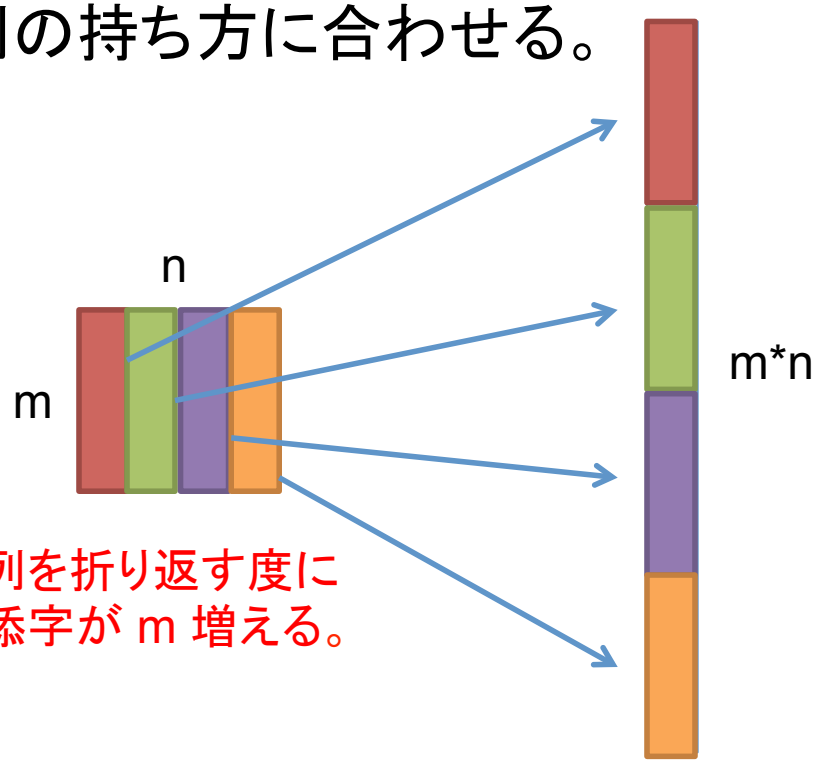
    特に、Fortran用の数値計算ライブラリを利用するときに有用。
- デメリット
  - 添字の取り扱いが若干複雑になる。

# 2次元配列から1次元配列への変換(2)

- 数値線形代数用の数値計算ライブラリは、元々、Fortranで作成されていたので、Fortranの配列の持ち方に合わせる。
- Aが m x n 行列のとき

  C言語: A[j][i] ⇔ A[i+j*m]

  Fortran: A(i,j) ⇔ A(i+(j-1)*m)



# 演習問題(1)

1. これまでに作成した2次元配列を用いた行列積の関数を１次元配列を用いたものに変更してみよう。
   - Fortranと同様の配列の持ち方に変更しよう。
   - A[i][k]やB[k][j]がどうなるかを考えよう。
   - 配列A, B, Cに対応する１次元配列は、malloc関数でメモリを確保しよう。
   - 最初は、アンローリングやブロック化をしていない単純なもので試そう。
   - 2次元配列を利用した場合と同じ結果になるか確かめよう。

2. 次に、配列の先頭のポインタを関数に渡せるように変更しよう。
   - 具体的には

   void multmm_p(int m, int p, int n, double *A, double *B, double *C);

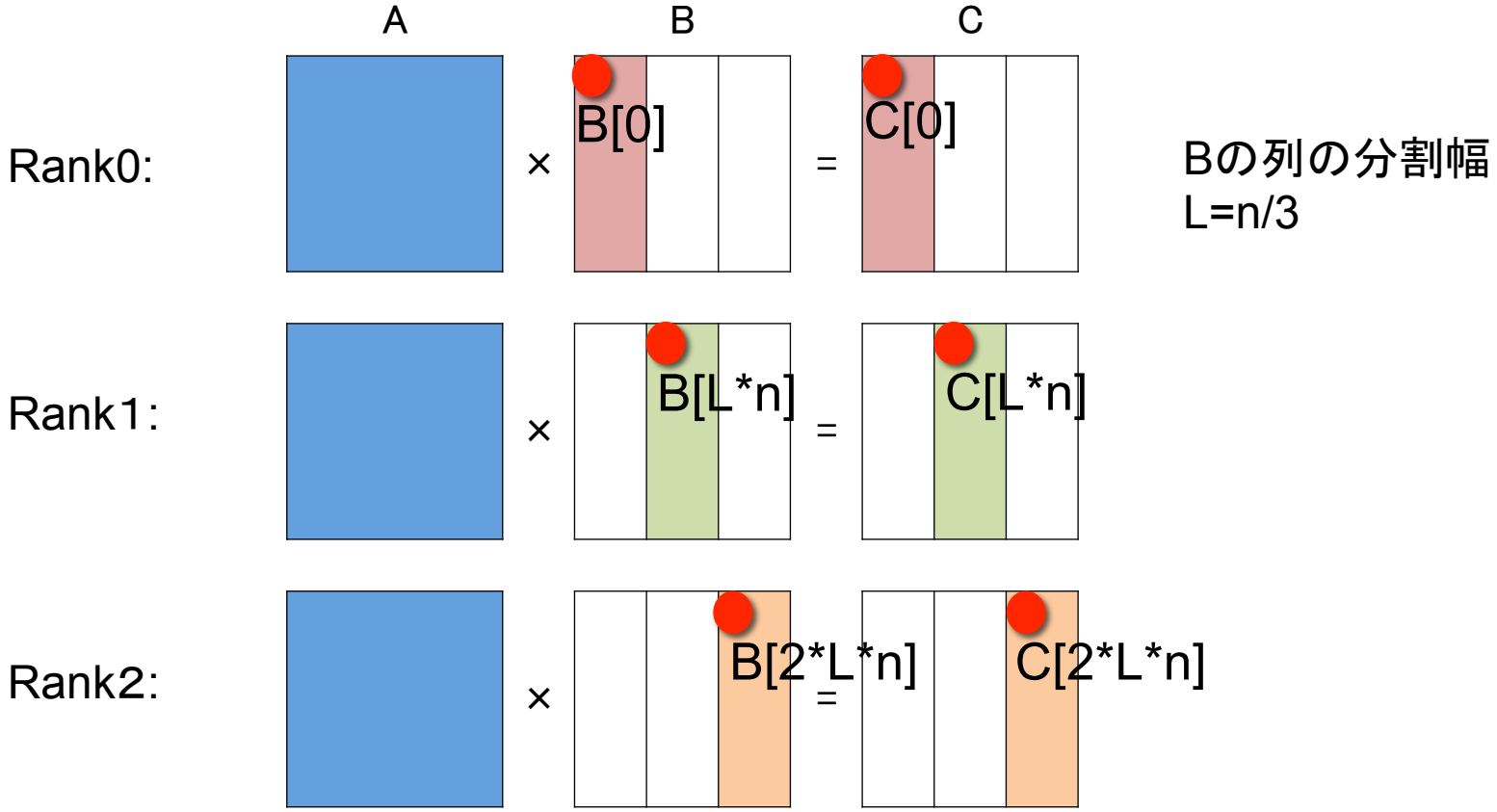
   のように関数を定義しよう（関数名は何でも良いです）。

# 演習問題(2)

1. 演習問題(1)で作成した1次元配列版の行列積プログラムを、アンローリングやブロック化で最適化してみよう。

# MPIを用いた行列積

- n x n行列A, Bについて、行列全体を各プロセスに送信。
- 各プロセスは、マスター側から送られて来たデータを用いて、担当部分の行列積を計算。



# 演習：MPIを用いた行列積

- 下記を参考に、MPIによる行列積プログラムを作成しよう（次ページもあり）。

```
int L, *np, *ptr;
ptr = malloc(sizeof(int)*p);
np = malloc(sizeof(int)*p);
L = n/p;
for (i = 0; i < p; i++) {
    ptr[i] = i*n*L;
    if (i == p - 1) {  /* (p – 1)番目のプロセスのみ */
        np[i] = L + n%p;  /* 余りも受けもつ */
    } else {
        np[i] = L;
    }
}
```

p: プロセス数
ptr[i]: i番目のプロセスが担当する行列の先頭の位置
np[i]: i番目のプロセスが担当する列数

# 演習：MPIを用いた行列積（つづき）

```
MPI_Bcast(A, n*n, MPI_DOUBLE, 0, MPI_COMM_WORLD);   /* 行列Aの送信 */
MPI_Bcast(B, n*n, MPI_DOUBLE, 0, MPI_COMM_WORLD);   /* 行列Bの送信 */

multmm_p(n, n, np[my_rank], A, &B[ptr[my_rank]], &C[ptr[my_rank]]);  /* 行列積 */

if (my_rank != 0) {
    tag = 100 + my_rank;    /* tagは適当に設定 */
    MPI_Send(&C[ptr[my_rank]], n*np[my_rank], MPI_DOUBLE, 0, tag,
    MPI_COMM_WORLD);    /* 行列Cの一部の送信 */
} else {
    for (rank = 1; rank < p; rank++) {
        tag = 100 + rank;
        MPI_Recv(&C[ptr[rank]], n*np[rank], MPI_DOUBLE, rank, tag,
        MPI_COMM_WORLD, &status); /* 行列Cの一部の受信 */
    }
}
```