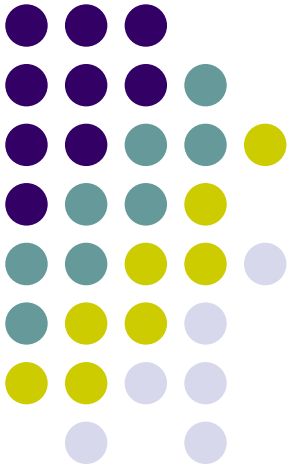
2013年度
実践的並列コンピューティング
第8回



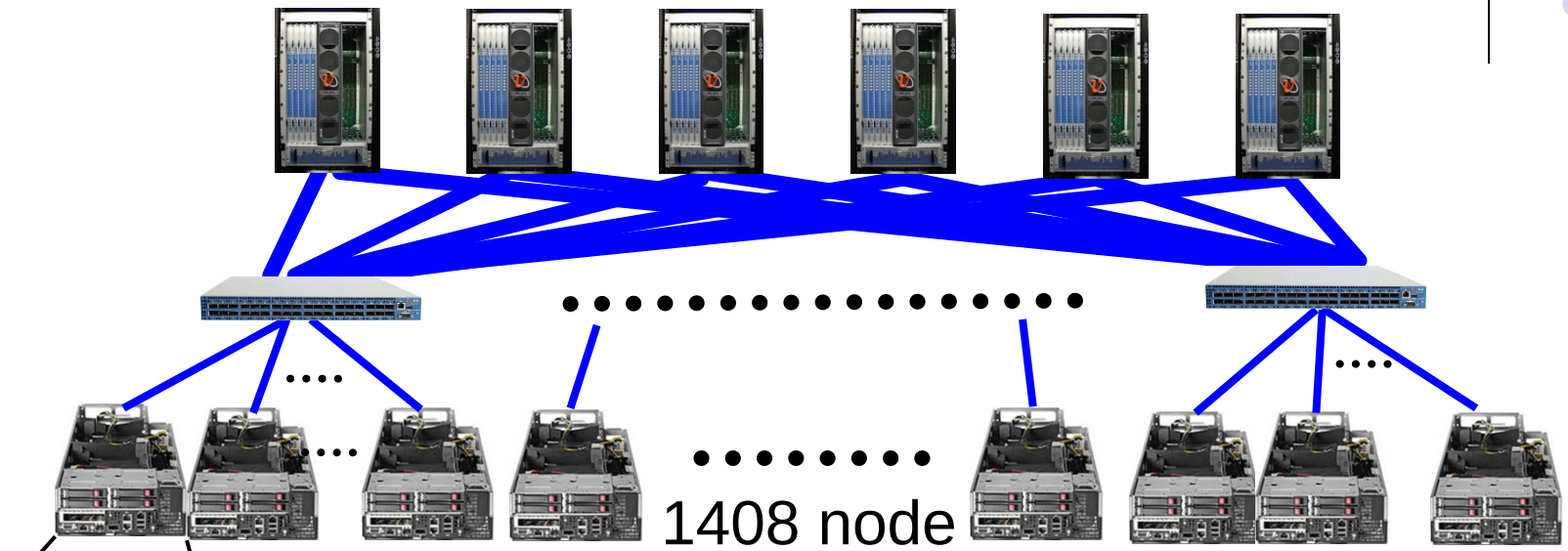
MPIによる
分散メモリ並列プログラミング(1)

遠藤 敏夫

endo@is.titech.ac.jp

2013年5月26日

スパコンシステムの多数の計算ノードを活用するには？



12core
+54GB

OpenMP

OpenMPは1ノードの中のCPUコアだけを使う

多数の計算ノードを活用するには？

1. (役割のそれぞれ違う)複数ジョブをバッチキューシステムに投入
   - パラメータをそれぞれ変えて投入することをパラメータスイープと呼ぶ
   - ジョブは独立に動き、原則的に協調しない

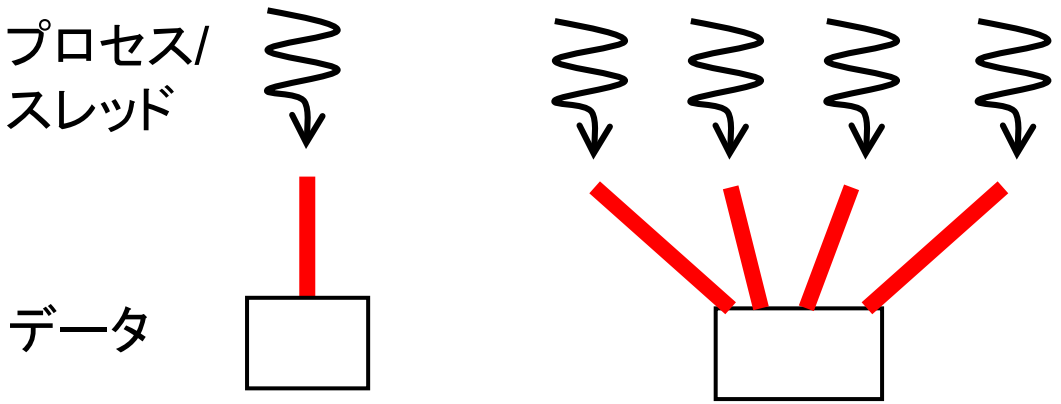
2. 一つのジョブが複数ノードを使いたい時には、分散メモリプログラミングを用いる
   - MPIやHadoop
   - Hadoopは、プロセス間の協調パターンが、Map-Reduceというパターンに限られる

並列プログラミングモデルの，メモリモデルによる分類

逐次

共有メモリモデル

分散メモリモデル

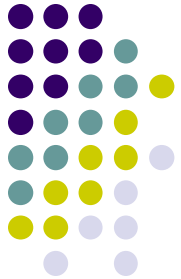


スレッド達が共通のデータにアクセス可能
- OpenMP(言語拡張)
- pthread(ライブラリ)
- CUDA (言語拡張)

プロセス間では通信が必要
- MPI (ライブラリ)
- socket (ライブラリ)
- Hadoop (フレームワーク)
- UPC, HPF, Chapel, X10

青字のものは，PGASと呼ばれ、一部共有メモリの概念あり
(partitioned global address space)

MPI(message-passing interface)とは



- 分散メモリ並列プログラミングの規格
- C, C++, Fortranに対応
- メッセージパッシングのためのライブラリ
- SPMDモデル．プロセス間の相互作用はメッセージで
  - MPI-2規格では，さらにRMA(remote memory access)が追加

科学技術演算でメジャーなMPI

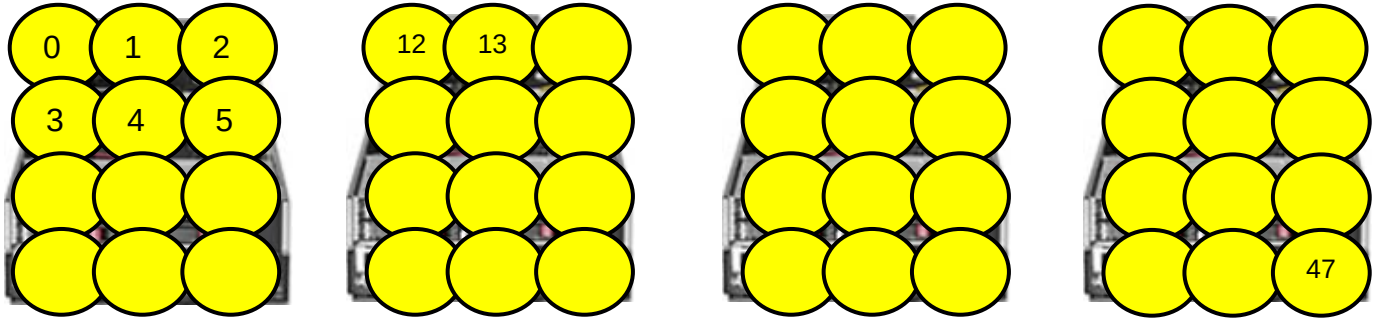
京スパコン上で稼働中のソフトウェア(一部)

ソフトウェア名	説明	並列化方法
feram	強誘電体MD	OpenMP
STATE	第一原理MD	MPI+OpenMP
FFVC	差分非圧縮熱流体	MPI+OpenMP
GT5D	5次元プラズマ乱流	MPI+OpenMP
FrontFlow/blue	有限要素法非圧縮熱流体	MPI+京並列コンパイラ
OpenFMO	FMO第一原理計算	MPI+OpenMP
pSpatiocyte	細胞内シグナル伝播計算	MPI+OpenMP
NEURON_K+	神経回路シミュレーション	MPI+OpenMP
SiGN-L1	遺伝子ネットワーク推定	MPI+OpenMP
NTChem/RI-MP2	電子相関計算	MPI+OpenMP+並列BLAS
HPL	Linpackベンチマーク	MPI+並列BLAS

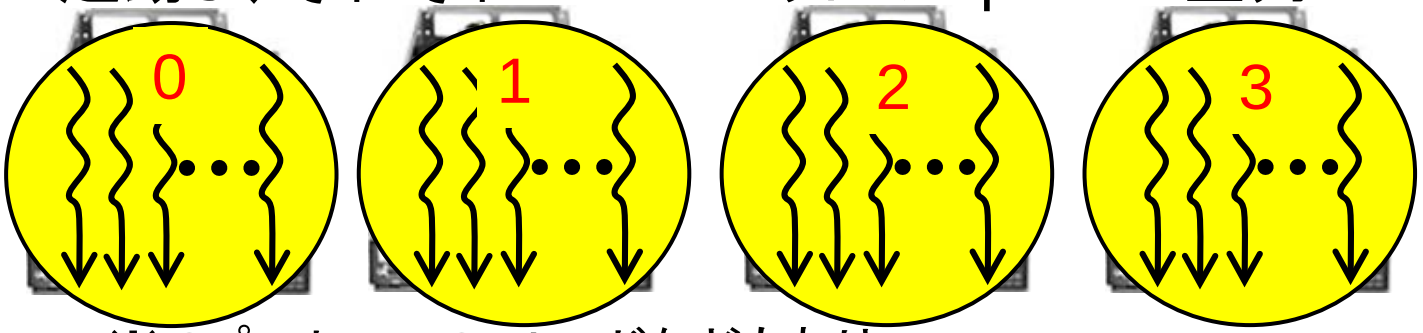
MPI+OpenMPとは？

- TSUBAMEでは1ノード12コア、京では1ノード8コアある。それを有効利用するには？

1. MPIのみ使う。図では48プロセス起動



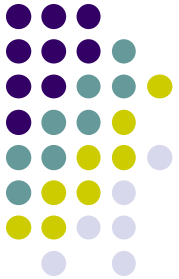
2. MPI+OpenMP(ハイブリッド並列)。図では4プロセス起動し、それぞれが12スレッドのOpenMP並列



※ 8プロセス×3スレッドなどもあり

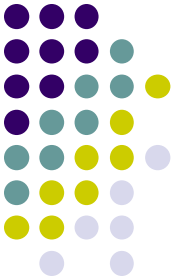
※ 1.より性能高い傾向にあるがプログラミング大変

OpenMPとMPI



- OpenMP
  - 共有メモリモデル
  - スレッド間のデータ移動は共有変数で
  - 排他制御によりrace conditionを防ぐ
  - 利用可能な並列度はノード内(TSUBAME2では12CPUコア)
  - #pragmaを無視すると逐次プログラムとして動作する場合が多い
- MPI
  - 分散メモリモデル
  - プロセス間のデータ移動はメッセージで
  - Critical sectionの代わりにメッセージで同期
  - 利用可能な並列度はノードを超える(TSUBAME2では10000CPUコア以上)
  - 逐次プログラムを基にする場合、全体の構造への大幅な変更が必要になりがち

MPIプロセスとメモリ



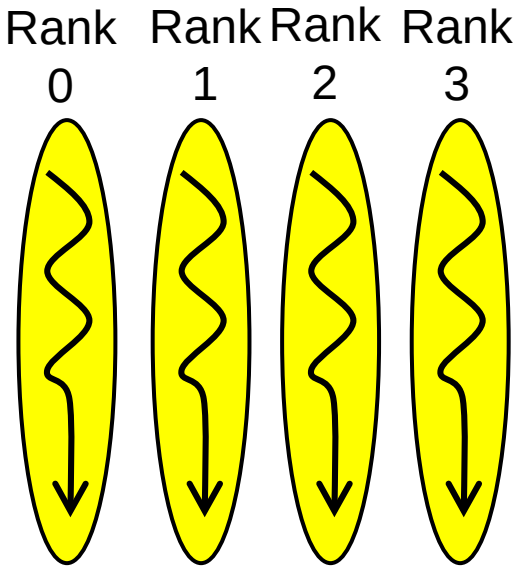
- 複数のプロセスが同一プログラムを実行(SPMDモデル)
- プロセスごとに別のメモリ空間 → 全ての変数(大域変数・局所変数)は各プロセスで別々
- プロセスには，0, 1, 2・・・という番号(rank)がつく
  - MPI_Comm_rank(MPI_COMM_WORLD, &rank); ランク取得
  - MPI_Comm_size(MPI_COMM_WORLD, &size); 全プロセス数取得
  - 0 ≦ rank < size
  - MPI_COMM_WORLDは，「全プロセスを含むプロセス集団(=コミュニケータ) 」
  - メッセージの送信先，受信元としてrankを利用

MPIプログラムの概要

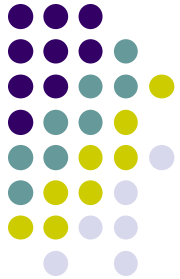
```
#include <stdio.h>
#include <mpi.h>

int main(int argc, char *argv[])
{
   MPI_Init(&argc, &argv);
```
← MPIの初期化

（計算・通信）

```
   MPI_Finalize();
}
```
← MPIの終了



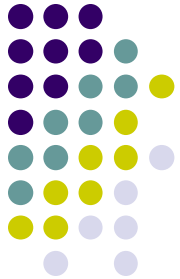
TSUBAME2上でのMPIプログラムのコンパイル



- mpiccというコマンドが使えることを確認する
  - デフォルトではOpenMPIというMPI処理系が設定されているはず
  - MVAPICHという別処理系も使える →詳細はTSUBAME2.5利用の手引き 5.1.4章

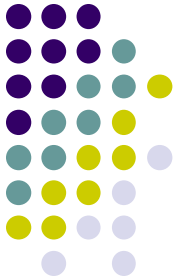
- コンパイル:
  - mpiccでコンパイル. オプションは一般のコンパイラと同じ
  - サンプルプログラムは, ~endo-t-ac/ppcomp/14/ 以下のmpitest1, pi-mpi, mm-mpi ディレクトリ
  - サンプルプログラムのコンパイルはmakeコマンドでok

TSUBAME2上でのMPIプログラムの実行 (1)



- インタラクティブノードでプログラムを実行する場合
  - mpirun –np [プロセス数] [プログラム名] [オプション]
  - プロセス数は4まで．かつ，実行時間は数分以内にとどめること
  - もっと大きいのはバッチキューでおねがいします

TSUBAME2上でのMPIプログラムのコンパイル，実行 (2)



- バッチキューを用い、testというMPIプログラムを，ノードあたり12プロセス×4ノード = 48プロセスで実行する場合

(1) スクリプトファイルの作成 (たとえばjob.shというファイル名):

```
#!/bin/sh
cd $PBS_O_WORKDIR
mpirun –np 48 –hostfile $PBS_NODEFILE ./myprog
```

全プロセス数。(2)とつじつまを合わせること

(1-2) chmod 755 job.sh コマンドで，「実行可能ファイル」にしておく

(2) t2subコマンドでジョブ投入

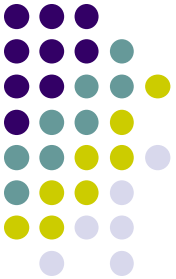
本授業の場合

```
t2sub –q S –W group_list=t2g-ppcomp
      –l select=4:mpiprocs=12 -l place=scatter ./job.sh
```

ノード数

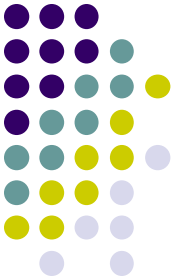
ノードあたりプロセス数

バッチキューを用いた場合の実行結果確認



- 以下は第二回でやった通り
  - ジョブが実行開始されたか、待ち中か確認
    - t2statコマンド
  - ジョブがprintfなどで出力した内容確認
    - ジョブが終了すると以下のファイルができるはず
      - OTHERS.oXXXX → 標準出力(stdout)の内容
      - OTHERS.eXXXX → 標準エラー出力(stderr)の内容＋バッチキューが出すシステムメッセージ

TSUBAME2上のキュー



- 用途に合わせて、いくつかキューが用意されている

※ キュー≒ノードの集合

※ t2subで-q S, -q Uのように指定

※ 原則、一ノード内に複数ジョブは混ざらない (V除く)

- Sキュー:
  - オールマイティ、MPIを使うときはこれ
  - ジョブあたり64ノードくらいまで行ける(混み具合にもよる)
  - ノードあたり12CPUコア、54GBメモリ、3GPU
- Uキュー:
  - ノード内並列向け。パラメータスイープに便利、安い
  - ノードあたり8CPUコア、32GBメモリ
- Gキュー:
  - GPUジョブ向け、安い
  - ノードあたり4CPUコア、12GBメモリ、3GPU
- Vキュー:コア単位のパラメータスイープ向け(授業グループでは使えない)

MPIの基本中の基本: メッセージの送信・受信

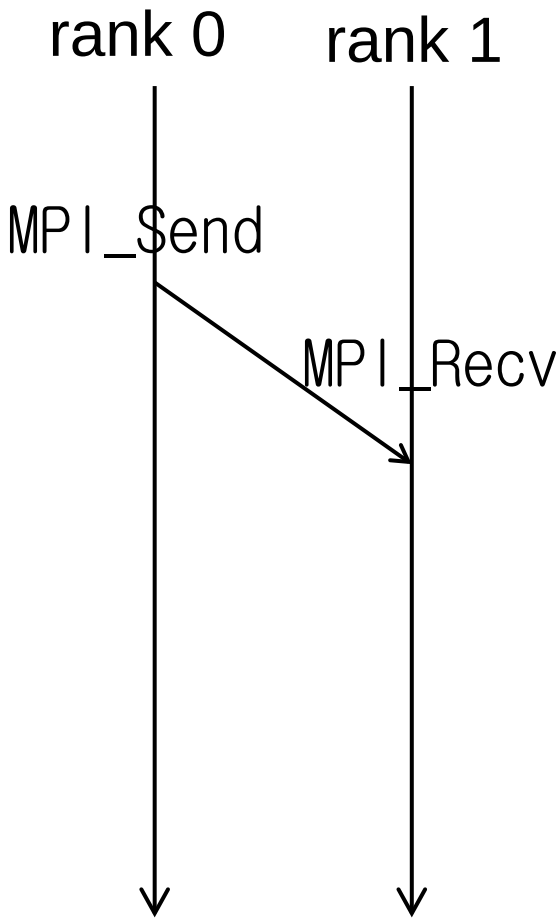
rank 0からrank1へ, int a[16]の中身を送りたい場合

- rank0側で

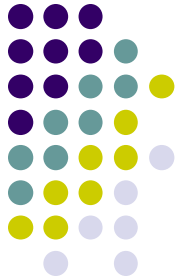
```
MPI_Send(a, 16, MPI_INT, 1,
    100, MPI_COMM_WORLD);
```

- rank1側で

```
MPI_Recv(b, 16, MPI_INT, 0,
    100, MPI_COMM_WORLD, &stat);
```



MPI_Send



```
MPI_Send(a, 16, MPI_INT, 1, 100, MPI_COMM_WORLD);
```

- a: メッセージとして送りたいメモリ領域の先頭アドレス
- 16: 送りたいデータ個数
- MPI_INT: 送りたいデータ型
  - 他にはMPI_CHAR, MPI_LONG. MPI_DOUBLE, MPI_BYTE・・・
- 1: メッセージの宛先プロセスのrank
- 100: メッセージにつけるタグ(整数)
- MPI_COMM_WORLD: コミュニケータ

MPI_Recv

```
MPI_Status stat;
MPI_Recv(b, 16, MPI_INT, 0, 100, MPI_COMM_WORLD, &stat);
```

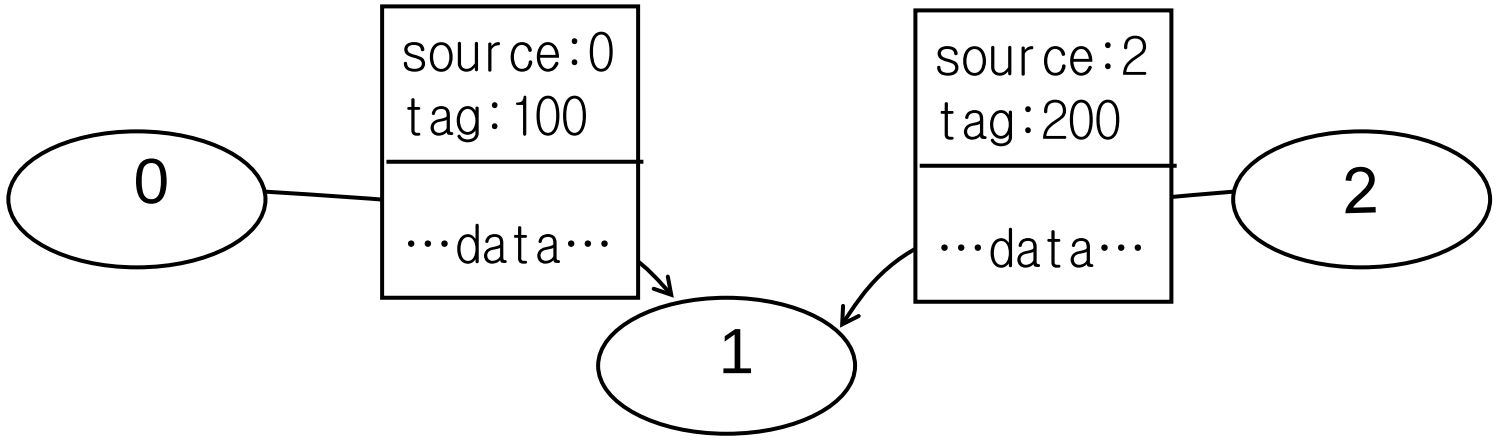
- b: メッセージを受け取るメモリ領域の先頭アドレス
  - 十分な領域を確保しておくこと
- 16: 受け取るデータ個数
- MPI_INT: 受け取るデータ型
- 0: 受け取りたいメッセージの送信元プロセスのrank
- 100: 受け取りたいメッセージのタグ．ユーザが決める整数
  - MPI_Sendで指定したものと同じなら受け取れる
- MPI_COMM_WORLD: コミュニケータ
- &stat: メッセージに関する補足情報が受け取れる

MPI_Recvを呼ぶと，メッセージが到着するまで待たされる（ブロッキング）

MPI_Recvのマッチング処理

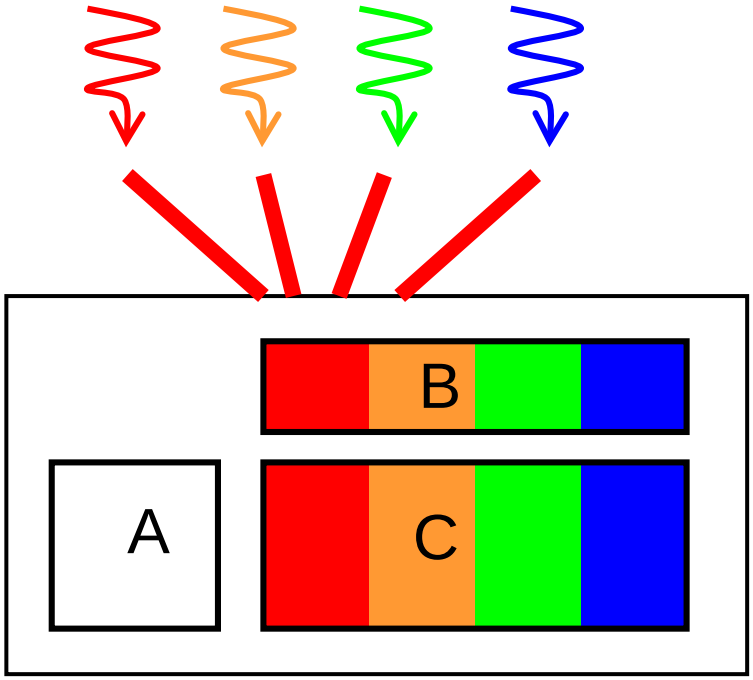
受信側には複数メッセージがやってくるかも → 受け取りたい条件を指定する

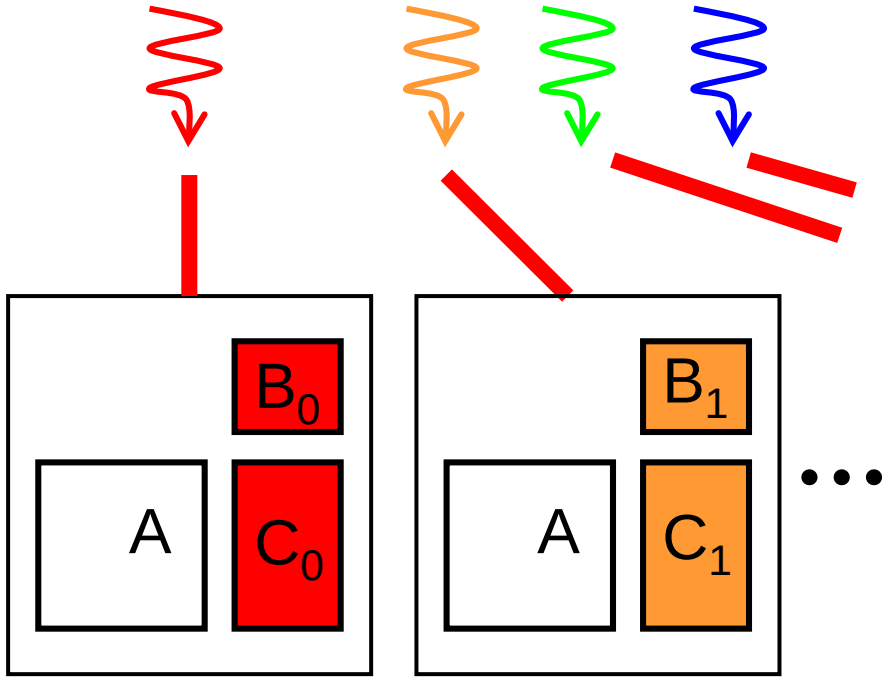
- 受け取りたい送信元を指定するか，MPI_ANY_SOURCE (誰からでもよい)
- 受け取りたいタグを指定するか，MPI_ANY_TAG(どのタグでもよい)



分散メモリと共有メモリの違い
行列積(C=AxB)の例

- 共有メモリ：計算をどうスレッドに分割するか
- 分散メモリ：計算とデータをどうプロセスに分割するか



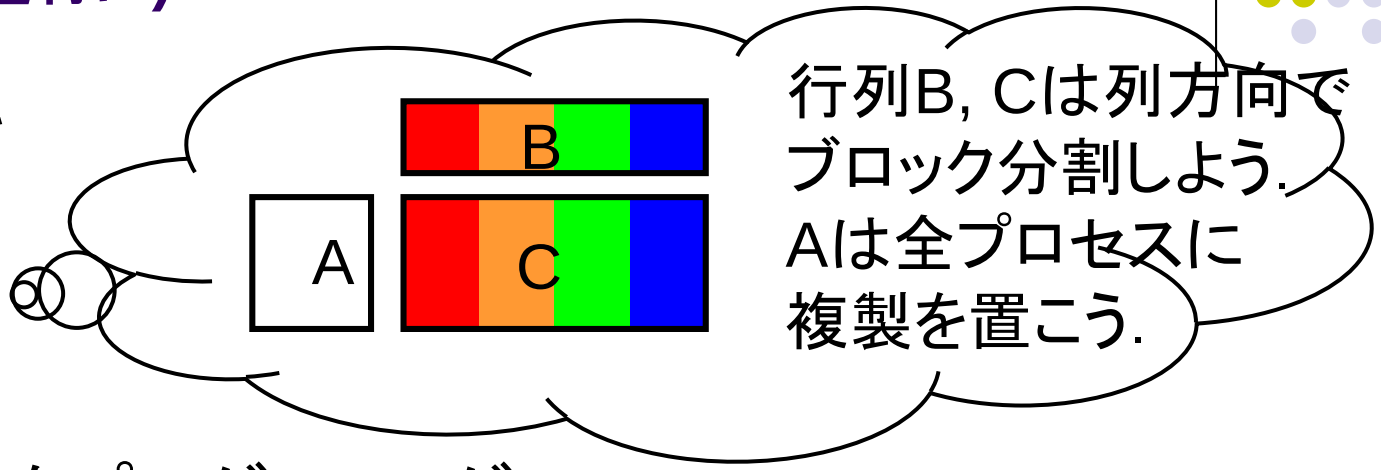


行列Aは全スレッドによってアクセスされる

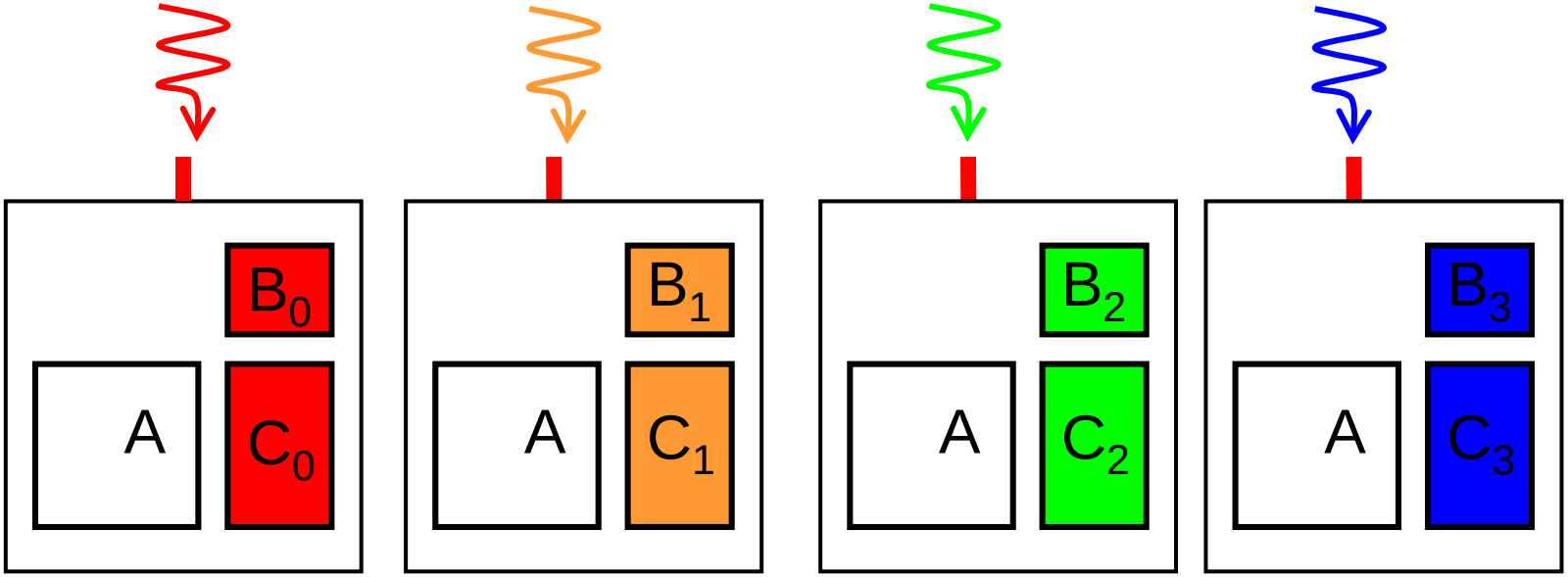
行列Aは全プロセスに置かれる
→ 後に改良案

分散メモリプログラミングとデータ配置
(mm-mpiを題材に)

配置方法を
決める:



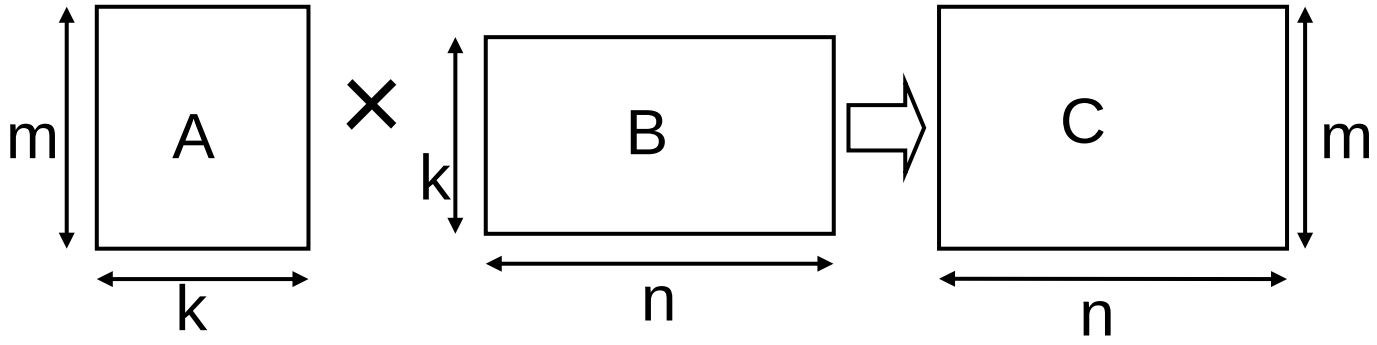
実際の配置をプログラミング:



行列積サンプルプログラム (再掲)

(m×k)行列と(k×n)行列の積

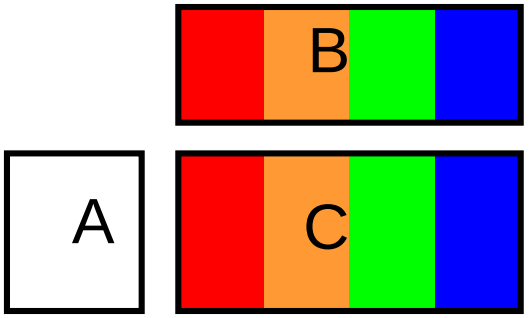
- 三重のforループで記述
- 動的長さ配列. 二次元を一次元で表現 (column-major)
- 実行オプション: ./mm [m] [n] [k]
- 計算量:O(m×n×k)



データ配置を考える

1. 初期配置
   - 各データは最初から分散しているとしてよいか？
   - 初期値はプロセス0が持っているとするか？
2. 計算中の配置
   - プロセス間の通信量が少なくなる配置が望ましい
   - メモリ消費とトレードオフの場合も
3. 結果の配置
   - 結果を(たとえばプロセス0に)集める必要はあるか？

(mm-mpiでは分割したままとした)

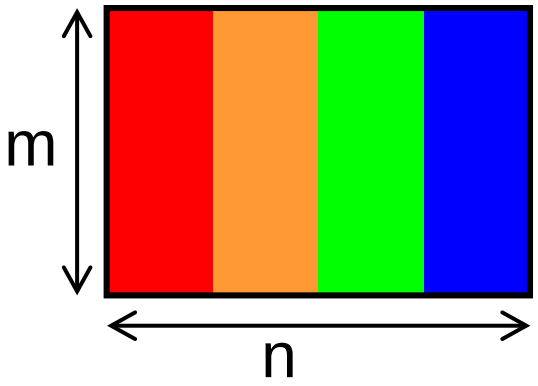


Cを列方向ブロック分割
⇒ Bも列方向の分割
　Aは全複製

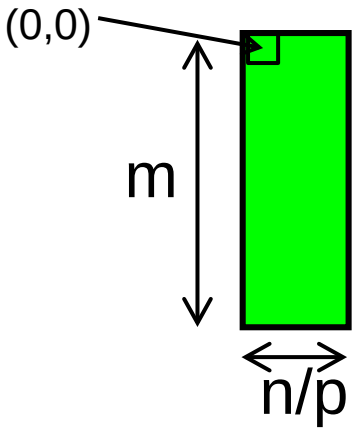
データ分散とプログラミング

- m×n行列をpプロセスで分割するとはどういうことか？
  - データ並びはcolumn-majorとする
  - ここでは割り切れる場合を仮定
- 各プロセスが持つのは，m×(n/p)の部分行列
  - m*(n/p)*sizeof(データ型)のサイズの領域をmallocすることに
  - 部分行列と全体行列の対応を意識する必要
    - プロセスrの部分行列の(i,j)要素⇔全体行列の(i, n/p*r + j)要素に対応

全体のイメージ



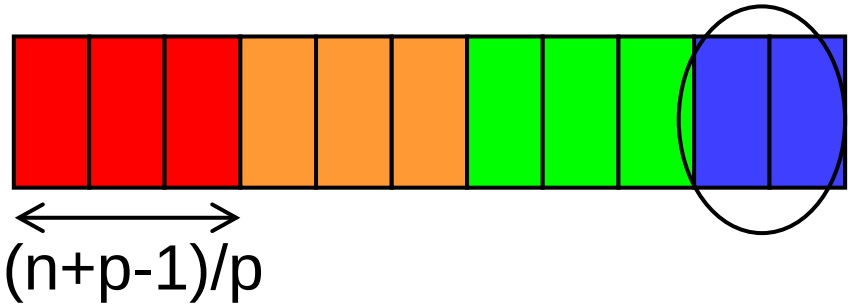
プロセスが持つ領域



意外とめんどうな端数処理

- データサイズnが，プロセス数pで割り切れるとは限らない
- 11個(11列)のデータを4プロセスで分割するには？
  - C言語の整数割り算は切り捨て
  - n/p = 2個ずつ担当していくとデータが余る → 切り上げの必要

→ (n+p-1)/p = 3個ずつ担当する.
最後のプロセスは他より仕事が少なくなる

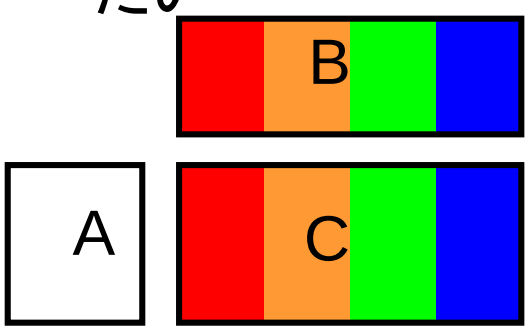


サンプルプログラムのdivide_length()関数は，自分の
担当場所の始点インデックスsと終点インデックスeを返す.
s以上e未満の，(e-s)個(列)のデータを担当することを示す.

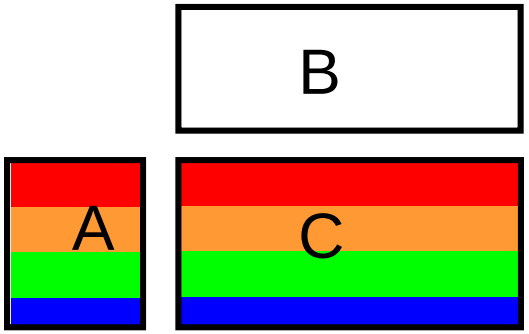
データ配置の再検討

- $C_{i,j}$の計算には，Aの第i列とBの第j列が必要

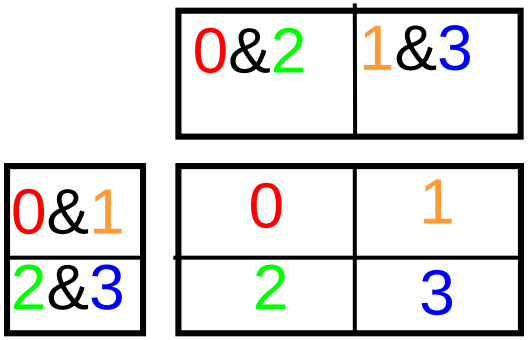
⇒　単純には、依存するデータをできるだけ同じプロセスに置きたい



Cを列方向ブロック分割
⇒ Bも列方向の分割
Aは全複製
(mm-mpiの方法)
O(mkp+nk+mn)



Cを行方向ブロック分割
⇒ Aも行方向の分割
Bは全複製
O(mk+nkp+mn)

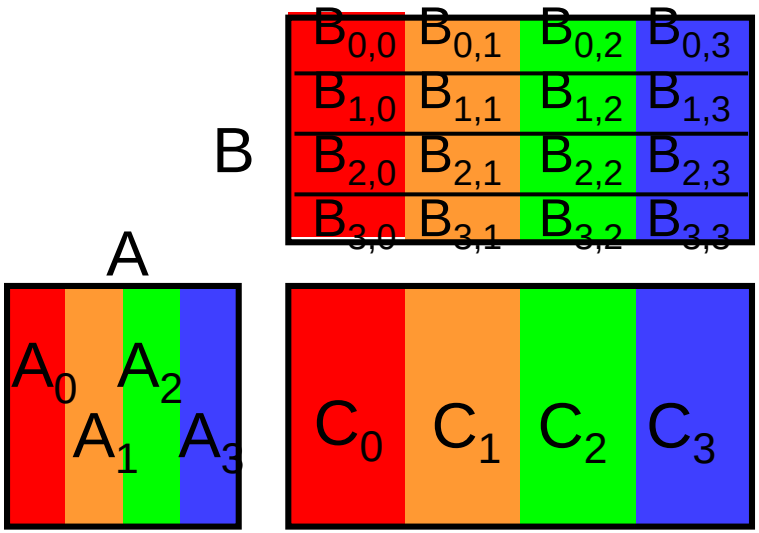


Cを二次元ブロック分割
⇒ A:行方向分割+複製
B:列方向分割+複製
O($mkp^{1/2}+nkp^{1/2}+mn$)

↑メモリ量の全プロセス合計

以上は全て，計算最中の通信は不要だがメモリ利用量が多い
プロセス数・問題サイズが大規模になるとダメ

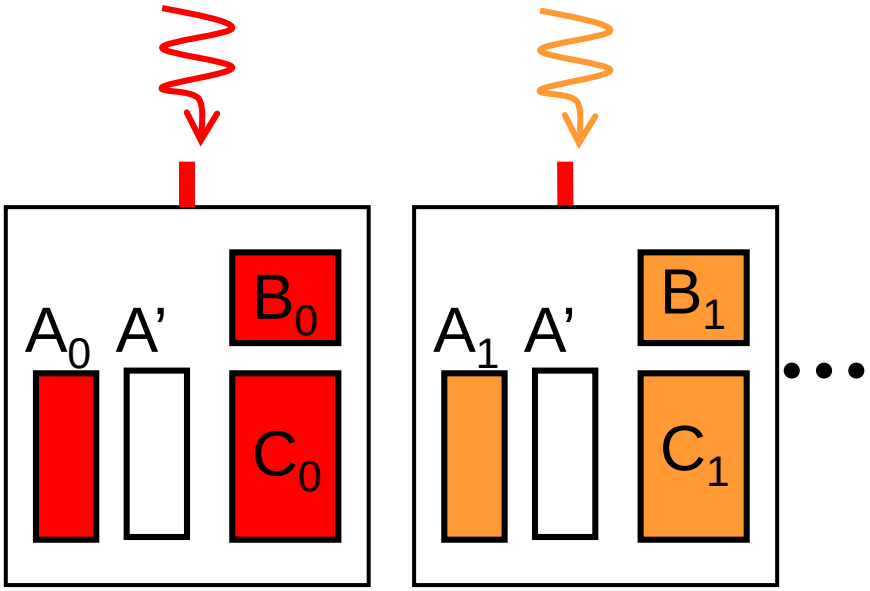
メモリ利用量を抑えるデータ配置の例



Aもブロック分割
(列方向でなくても可)
⇒ ローカルデータだけでは
C_iの計算できないので，
通信が必要

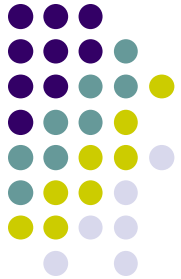
<u>第0フェーズ：</u>
プロセス0がA_0をBcast
各プロセスiは，
C_i += A_0 × $B_{0,i}$ を計算
<u>第1フェーズ：</u>
プロセス1がA_1をBcast
各プロセスiは，
C_i += A_1 × $B_{1,i}$ を計算
：
以下同様に，第(p-1)フェーズまで行う

実際のメモリ配置はどうなる？



- 各プロセスは，分割されたAに加え，受信バッファ(A')を用意する
- 第rフェーズでは，
  - プロセスrはAからA'へデータコピー　(省略する手法もあり)
  - プロセスrをルートとし，領域A'をMPI_Bcast

本授業のレポートについて

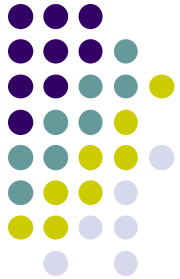


- 各パートで課題を出す。2つ以上のパートのレポート提出を必須とする

  予定パート:

  - OpenMPパート
  - MPIパート
  - GPUパート

MPIパート課題説明 (1)



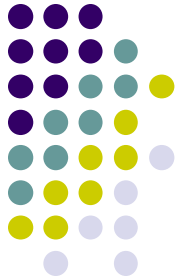
以下の[M1]—[M3]のどれか一つについてレポートを提出してください

[M1] diffusionサンプルプログラムを，MPIで並列化してください.

オプション:

- MPIで一般のサイズ(プロセス数で割り切れないかもしれない)に対応するには端数処理が必要である。本レポートではその対応はオプションとする
- より良いアルゴリズムにしてもよい．ブロック化・計算順序変更でキャッシュミスを減らせないか？

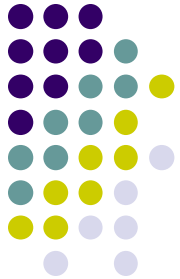
MPIパート課題説明/Report (2)



[M2] MPIで並列化され，メモリ利用量を抑えた行列積プログラムを実装してください

- mm-mpiサンプルの改造でよい
- データ分割は本授業の通りでもそれ以外でもよい
- 今回のスライドのアルゴリズムよりも進化した、SUMMA (Scalable Universal Matrix Multiplication Algorithm)[Van de Geijn 1997] もok
- 端数処理はあった方が望ましいが，必須ではない

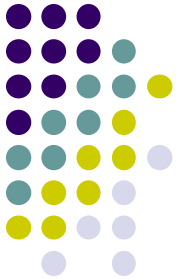
MPIパート課題説明/Report (3)



[3] 自由課題：任意のプログラムを，MPI(MPI-2も可)を用いて並列化してください．

- 単純な並列化で済む問題ではないことが望ましい
  - スレッド・プロセス間に依存関係がある
  - 均等分割ではうまくいかない、など
- たとえば，過去のSuperConの本選問題
  http://www.gsic.titech.ac.jp/supercon/
  たんぱく質類似度(2003)，N体問題(2001)・・・
  入力データは自分で作る必要あり
- たとえば，自分が研究している問題

課題の注意



- いずれの課題の場合も，レポートに以下を含むこと
  - 計算・データの割り当て手法の説明
  - TSUBAME2などで実行したときの性能
    - プロセッサ(コア)数を様々に変化させたとき．大規模のほうがよい．XXコア以上で発生する問題に触れているとなお良い
    - 問題サイズを様々に変化させたとき(可能な問題なら)
  - 高性能化のための工夫が含まれているとなお良い
    - 「XXXのためにXXXをしてみたが高速にならなかった」のような失敗でも可
  - 作成したプログラムについても、zipなどで圧縮して添付
    - 困難な場合，TSUBAME2の自分のホームディレクトリに置き，置き場所を連絡

課題の提出について

- MPIパート提出期限
  - 6/30（月）23:50
  - その前のOpenMPパート6/2にも注意
- OCW-i ウェブページから下記ファイルを提出のこと
- レポート形式
  - 本文：PDF, Word, テキストファイルのいずれか
  - プログラム：zip形式に圧縮するのがのぞましい

- OCW-iからの提出が困難な場合、メールでもok
  - 送り先： ppcomp@el.gsic.titech.ac.jp
  - メール題名： ppcomp report

次回: 6/2(月)

- MPI (2)

- スケジュールについてはOCW pageも参照
  - http://www.el.gsic.titech.ac.jp/~endo/

  → 2014年度前期情報(OCW) → 講義ノート