SuperCon 2010
の資料から

目次

２．ＭＰＩプログラミング入門

※ この資料は，スーパーコン10で使用したものである．ごく基本的な内容なので，現在でも十分利用できると思われるものなので，ここに紹介させて頂く．ただし，古い情報も含まれているので注意が必要である．
今年度版の解説は，本選の初日に配布する予定である．

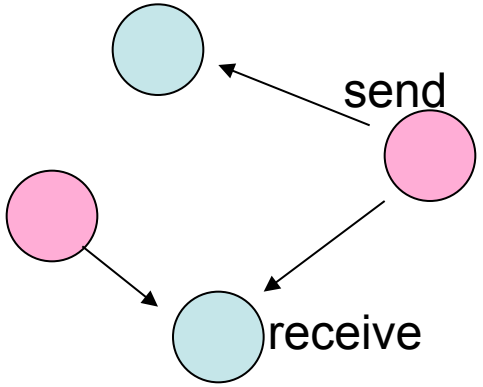
SuperCon 2010
の資料から

２．ＭＰＩプログラミング入門

(1) 基本

【説明】

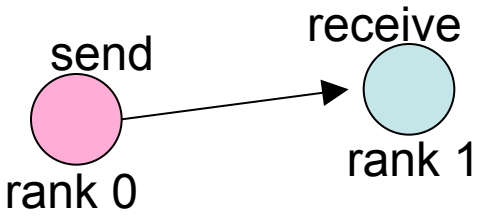
- ＭＰＩ（message passing interface）とは並列計算を記述するための基本命令を書く方法．ＭＰＩにより並列プログラムを記述することができる．といっても，そう多くの命令は必要なく，ほとんどの場合，ＭＰＩのごく基本的な命令を知っていれば並列プログラムを書くことはできる．今回はＣ言語の上で使うＭＰＩの基本命令を紹介する．
- ＭＰＩにより実現されるのは，同一のプログラムを複数のプロセスが実行する形の並列計算である．各プロセスは「名前」に相当するランク（rank）という番号を持っている．とくにルートプロセスのランクは 0 である．このランクを使って区別すれば，同じプログラムでも，異なる計算を実行することも可能．
- プロセス間でメッセージを送り合うことで，情報を交換する．メッセージは一般には 1 つの配列（文字や数の）として受け渡す．



(1) 基本

注）黒太字は「おまじない」

実例プログラム: hello.c




```
#include <mpi.h>
int
main(int argc, char *argv [ ]) {
   char msg[20];
   int myrank;
   MPI_Status status;  ← 状態を知るための変数 status の定義
   MPI_Init( &argc, &argv );  ← 初期化
   MPI_Comm_rank( MPI_COMM_WORLD, &myrank );
   if (myrank == 0) {
      strcpy(msg, "Hello, there");
      MPI_Send( msg, strlen(msg)+1, MPI_CHAR, 1, 99, MPI_COMM_WORLD);
      printf(“message %s sent to rank %d\n", msg, myrank);
   }
   else {
      MPI_Recv(msg, 20, MPI_CHAR, 0, 99, MPI_COMM_WORLD, &status);
      printf("rank %d received: %s\n", myrank, msg);
   }
   MPI_Finalize();   ←  終了
   return 0;
}
```

SuperCon 2010
の資料から

(1) 基本

・実行例
% mpicc hello.c　← プログラム hello.c のコンパイル
% mpirun -np 2 ./a.out　← 実行（プロセス数 2）

【補足説明】

・命令

MPI_Comm_rank(MPI_COMM_WORLD, &整数変数);
MPI_Send(変数 or 配列, データ長, 型, 送信先, 99, MPI_COMM_WORLD);
MPI_Recv(変数 or 配列, データ長, 型, 送信元, 99, MPI_COMM_WORLD, &status);
↑ MPI_ANY_SOURCE で任意の送信元を指定

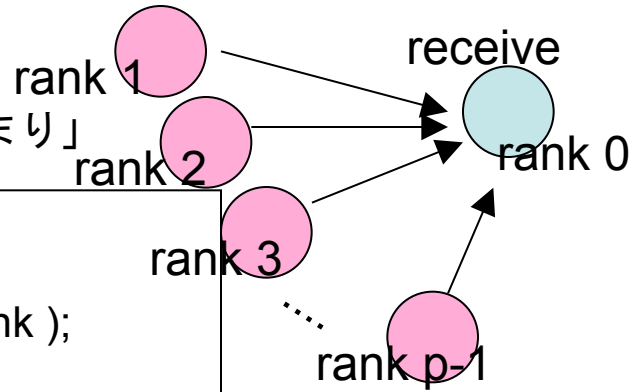
・ＭＰＩデータ型

MPI_CHAR char　　MPI_SHORT short
MPI_INT int　　MPI_LONG long
MPI_UNSIGNED_CHAR unsigned char
MPI_UNSIGNED_SHORT unsinged short
MPI_UNSIGNED unsinged int　　MPI_UNSIGNED_LONG unsinged long
MPI_FLOAT float　　MPI_DOUBLE double
MPI_LONG_DOUBLE long double

(1) 基本

実例プログラム: sum.c 並列和

```
#include <mpi.h>
int
main(int argc, char *argv [ ]) {    ↓以下でも「お決まり」
  int myrank, p, sum, v, i;
  MPI_Status status;
  MPI_Init( &argc, &argv );
  MPI_Comm_rank( MPI_COMM_WORLD, &myrank );
  MPI_Comm_size( MPI_COMM_WORLD, &p );
  if (myrank == 0) {
    sum = 0;
    for (i=1; i<p; i++) {
      MPI_Recv(&v, 1, MPI_INT, i, 99, MPI_COMM_WORLD, &status);
      sum = sum + v;
  } }
  else {
    MPI_Send(&myrank, 1, MPI_INT, 0, 99, MPI_COMM_WORLD);
  }
  if (myrank == 0) printf("sum of all ranks = %d\n", sum);
  MPI_Finalize();
  return 0;
}
```



SuperCon 2010
の資料から

(1) 基本

・実行例
% mpicc sum.c ← プログラム sum.c のコンパイル
% mpirun -np 4 ./a.out ← 実行(プロセス数 4)

【補足説明】
・命令
MPI_Comm_size(MPI_COMM_WORLD, &整数変数 **);**
実行しているプロセスの個数を得る

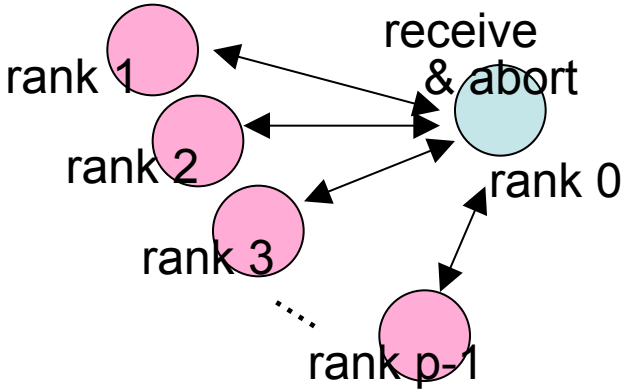
(2) 主従関係プログラム（次のページの例）

【補足説明】
・命令
MPI_Recv(&変数, 1, MPI_INT,
MPI_ANY_SOURCE, 88, MPI_COMM_WORLD, **&status**);
↑ **status.MPI_SOURCE** で送信元のランクを得る
MPI_Abort(MPI_COMM_WORLD, -1);
すべてのプロセスを強制終了させる

SuperCon 2010
の資料から

(2) 主従関係プログラム

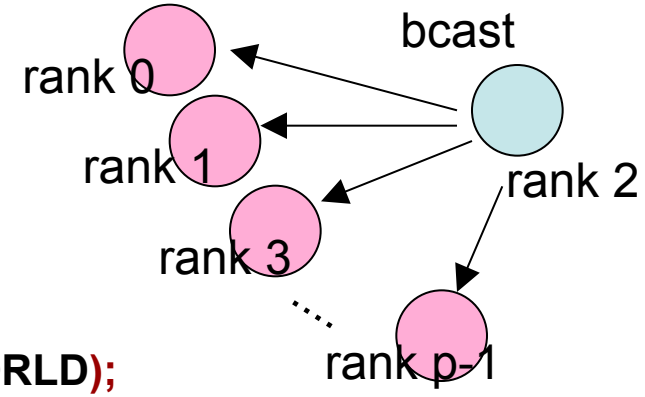
実例プログラム: master-worker.c




```
#include <mpi.h>
int
main(int argc, char *argv [ ]) {
   int myrank, p, n = 1;
        ～「お決まり」の4 行
   while(1) {
      if (myrank == 0) {
         MPI_Recv( &p, 1, MPI_INT, MPI_ANY_SOURCE, 88,
                                                MPI_COMM_WORLD, &status );
         printf("message from rank %d (= %d)\n", p, status.MPI_SOURCE);
         n++;
         if( n > 11 ) {  MPI_Abort( MPI_COMM_WORLD, -1 );  break; }
      }
      else {
         sleep(myrank);
         MPI_Send( &myrank, 1, MPI_INT, 0, 88, MPI_COMM_WORLD );
   } }
   MPI_Finalize();
   return 0;
}
```

(3) 集合通信系プログラム

実例プログラム: bcast.c



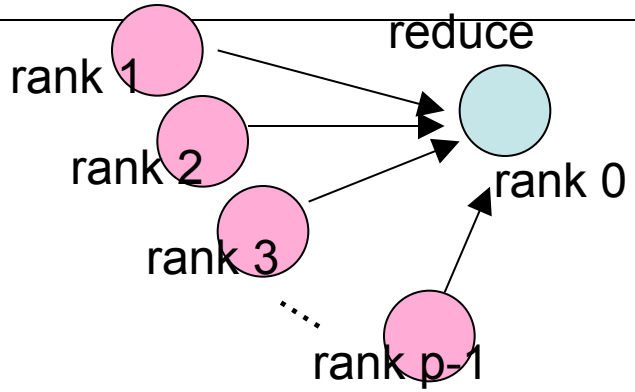

```
#include <mpi.h>
int
main(int argc, char *argv) {
   int myrank, p, sum, u, v, i;
        ≀「お決まり」の4 行
   if(myrank == 2)  v = 7;
   MPI_Bcast (&v, 1, MPI_INT, 2, MPI_COMM_WORLD);
   if (myrank == 0){
      sum = 0;
      for (i=1; i<p; i++){
         MPI_Recv(&u, 1, MPI_INT, i, 99, MPI_COMM_WORLD, &status);
        sum = sum + u;
   } }
   else {
      u = v * myrank;
      MPI_Send(&u, 1, MPI_INT, 0, 99, MPI_COMM_WORLD);
   }
   if (myrank == 0) printf("sum of all ranks*v = %d\n", sum);
   MPI_Finalize();
   return 0;
}
```

SuperCon 2010
の資料から

(3) 集合通信系プログラム

実例プログラム: reduce.c

```
#include <mpi.h>
int
main(int argc, char *argv [ ]) {
   int myrank, p, sum, v;
       ～ 「お決まり」の4 行
   v = myrank;
   MPI_Reduce(&v, &sum, 1, MPI_INT, MPI_SUM, 0, MPI_COMM_WORLD);
   if (myrank == 0) printf("sum of all ranks = %d\n", sum);
   MPI_Finalize();
   return 0;
}
```



(3) 集合通信系

【補足説明】

・命令

MPI_Bcast (変数など, データ長，型, **ルート**, **MPI_COMM_WORLD);**
ルートとして指定されたプロセスから
全プロセスの対応する変数にデータを送る

MPI_Reduce (変数, 答え用変数，データ長，型,
演算，ルート, MPI_COMM_WORLD);
ルートとして指定されたプロセスに対し
全プロセスの変数の値を元に「演算」した「答え」を送る

演算の種類

MPI_MAX（最大），**MPI_MIN**（最小），
MPI_SUM（合計），**MPI_PROD**（積），
MPI_LAND（論理AND），　**MPI_BAND**（ビットAND），
MPI_LOR（論理OR），　**MPI_BOR**（ビットOR），
MPI_MAXLOC（最大と位置），　**MPI_MINLOC**（最小と位置）

注）通常にプログラミングするより速い場合が多い