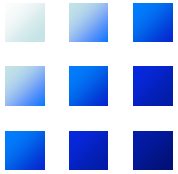


MPI.NETではじめる C#並列プログラミング

同志社大学大学院 知識工学専攻
博士後期課程2年 中尾昌広

2009年1月20日

目的

- MPI.NETの概要の説明
- MPI.NETとC#を使った並列プログラミングを紹介

目次

■ 並列計算のための準備
- MPI.NET、C#、MPI、実行環境の説明

■ MPI.NETで利用する基本的な関数の説明
- 1対1通信、集合通信など

■ MPI.NETとC#を用いたプログラム例
- Hello World、モンテカルロ法によるπ計算

MPI.NETとは

- .NET Framework上で動作する並列計算用ライブラリ
- .NETで用いられている言語の全て（特にC#）をサポート
- ノード間の通信を簡易に行えるAPIを提供

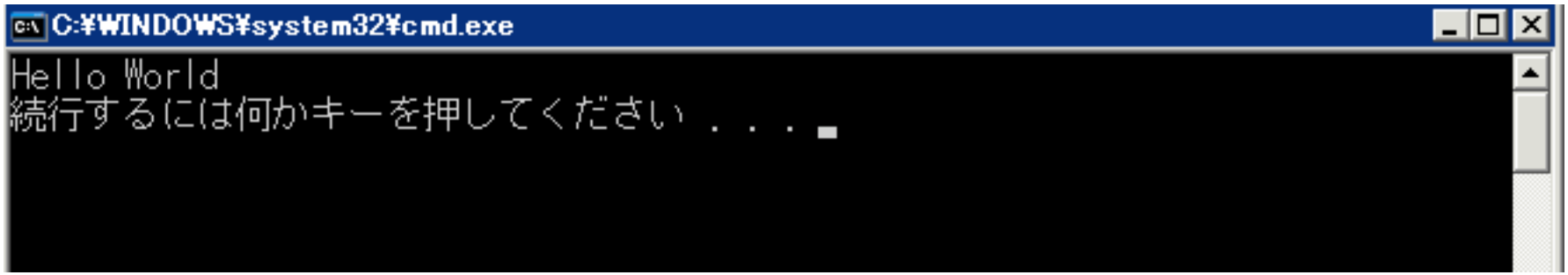
C#とは

- Microsoft社が開発したプログラミング言語
- .NET環境の中心的言語
- javaに似たオブジェクト指向型言語であり、
  プロセッサに依存しない実行ファイルを生成可能
- 他の.NET言語(Visual Basic .NETやVisual C++)と
  相互に連携可能。他言語で記述されたクラスを
  継承することも、その逆も可能

C#のプログラミング例

```
class Helloworld                    // クラスの宣言
{
  public static void Main()    // メイン関数
  {
    // コンソールにメッセージを出力
    System.Console.Write(“Hello World\n”);
  }
}
```

出力結果



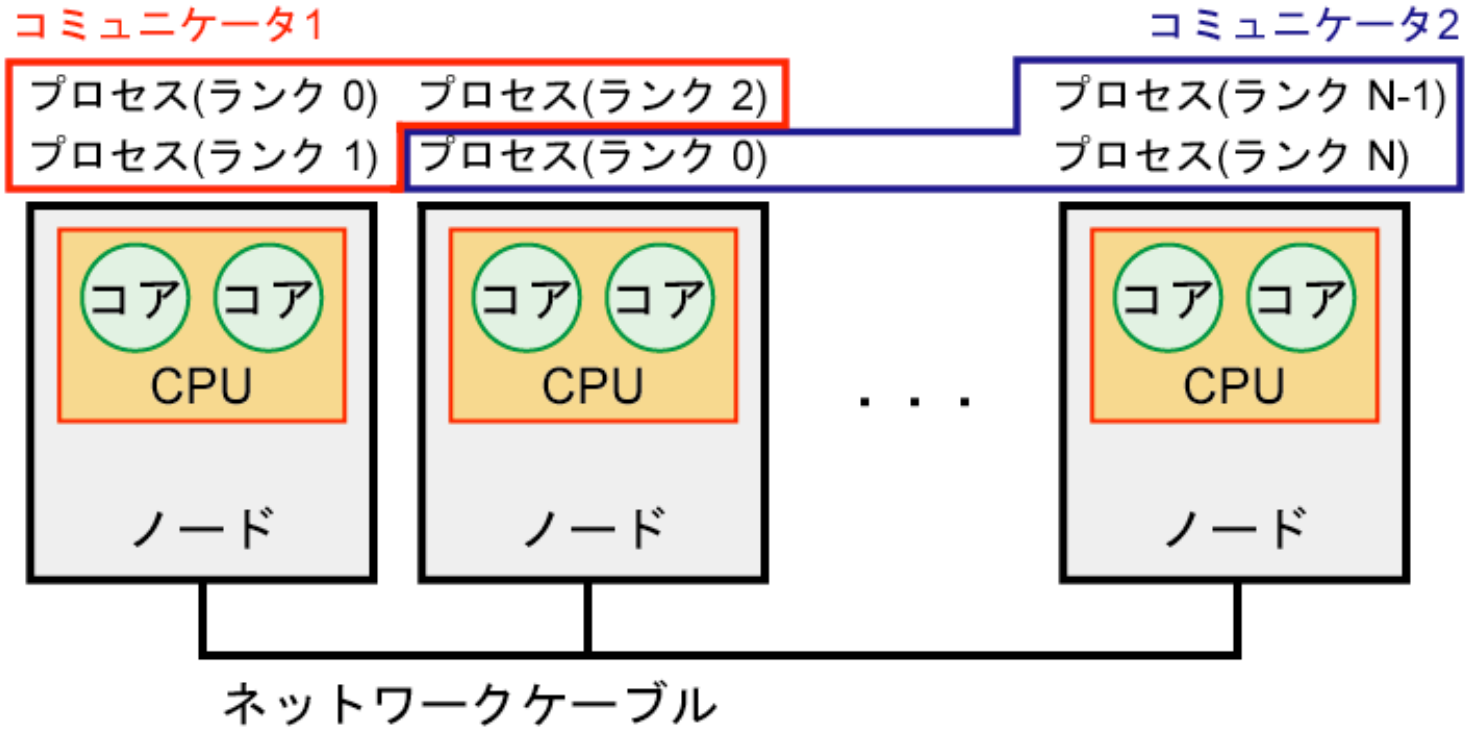
MPI（Message Passing Interface）とは

- 分散メモリプログラミングの標準規格
- 並列計算時にデータを通信するために用いるメッセージ操作の標準仕様
- MPIに従った実装が数多く存在する
- MPI-1とMPI-2があり、2の方が新しく機能も豊富

実装名	MPI.NET	MPICH	MPICH2	LAM/MPI	MS-MPI
対応Ver.	2	1	2	1	2
対応言語	.NET言語（C#など）	C, C++, fortran			

MPIの用語

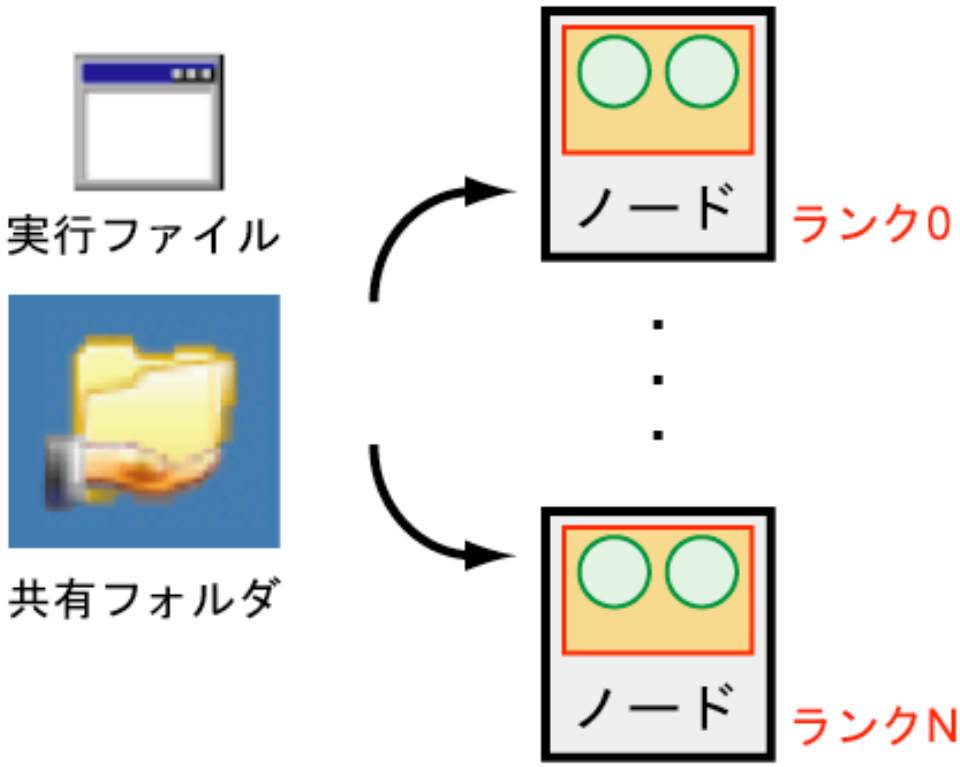
■ ノード　：マシン1台1台のこと
■ プロセス：プログラムの実行単位。それぞれのプロセスは
　　独立したメモリ空間を持つ
■ランク　　：各プロセスに付けられた名前（プロセスID）
■コミュニケータ：通信グループの単位



並列プログラム実行の流れ

- 複数のノードで動作させる実行ファイルは基本的に1つのみ
- if else文などを使って、ランクによって動作を変える

```
if (rank == 0){

  // ランク0にさせたい内容

} else if( rank == 1 ){

  // ランク1にさせたい内容

}

                .
                .
                .
```



開発環境について

緑字下線は今回用いた環境

■ C#の開発環境

Visual Studio 2008、Visual C# 2008 Express Edition

■ .NET環境

.NET Framework 3.5

■ MPI.NET

MPI.NET SDK 1.0

■ OSなど

Windows HPC Server 2008、HPC Pack 2008 SDK
Microsoft Compute Cluster Server 2003、
Microsoft Compute Cluster Pack SDK

MPI.NETの基本関数について

よく利用される通信関数

■1対1通信

（同期通信）Send、Receive

（非同期通信）ImmediateSend、 ImmediateReceive

■集合通信（多対多通信）

Barrier、Gather、Broadcast、Reduce

同期通信と非同期通信

■ 同期通信（ブロッキング通信）

・ 送信（受信）が終了するまでは、それ以降のプログラムは実行されない

・デットロックの可能性

・Send、Receive

■非同期通信（ノンブロッキング通信）

・通信処理はバックグラウンドで実行される

・ ImmediateSend、 ImmediateReceive

Send（同期型送信用関数）

public void Send<T>(value, dest, tag)

- value：送信したい値
- dest　：送信先（ランクを指定）
- tag　：データ識別用タグ（int型の整数）

```
if (comm.Rank == 0)
{
    string value = “Windows”;
    comm.Send(value, 1, 9);
}
```

Receive（同期型受信用関数）

public T Receive<T>(source, tag)

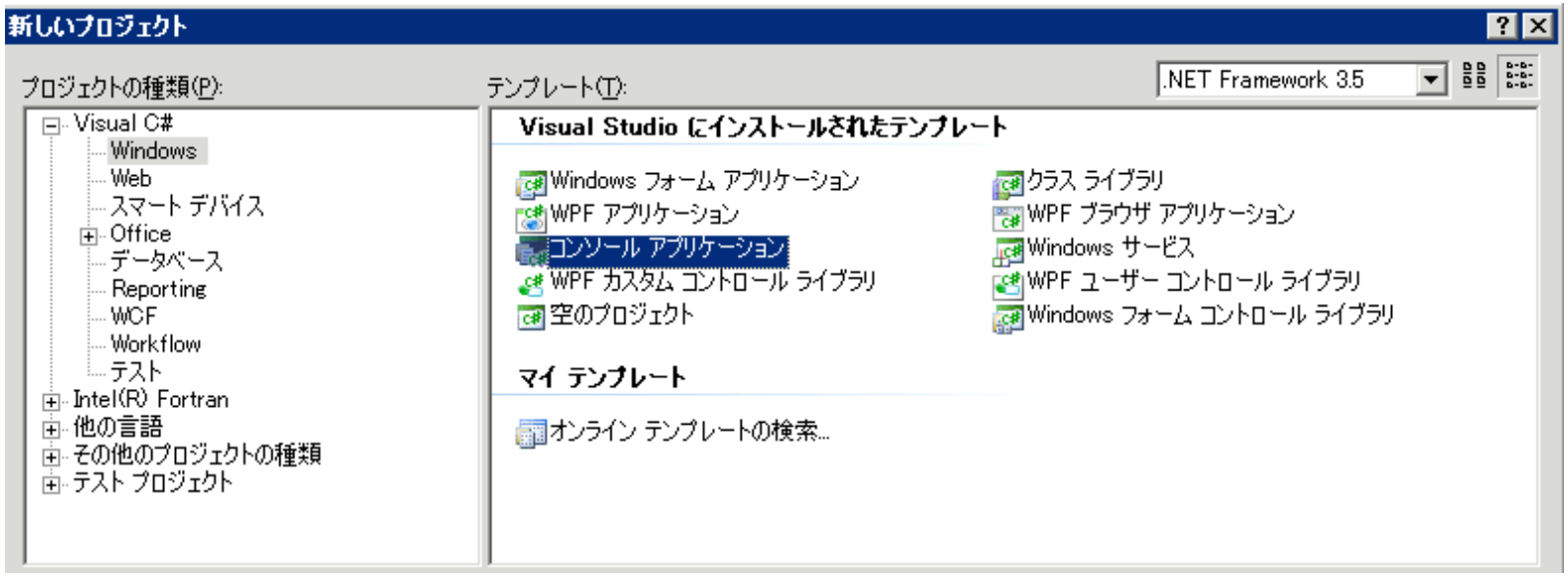
- source：送信元（ランクを指定）
- tag　：データ識別用タグ（int型の整数）
- 返り値が受信されたデータになる

```
if (comm.Rank == 0)
{
    string value = “Windows”;
    comm.Send(value, 1, 9);
}
```

```
else if(comm.Rank == 1)
{
   string msg =
     comm.Receive<string>(0, 9);

   Console.Write(msg);
}
```

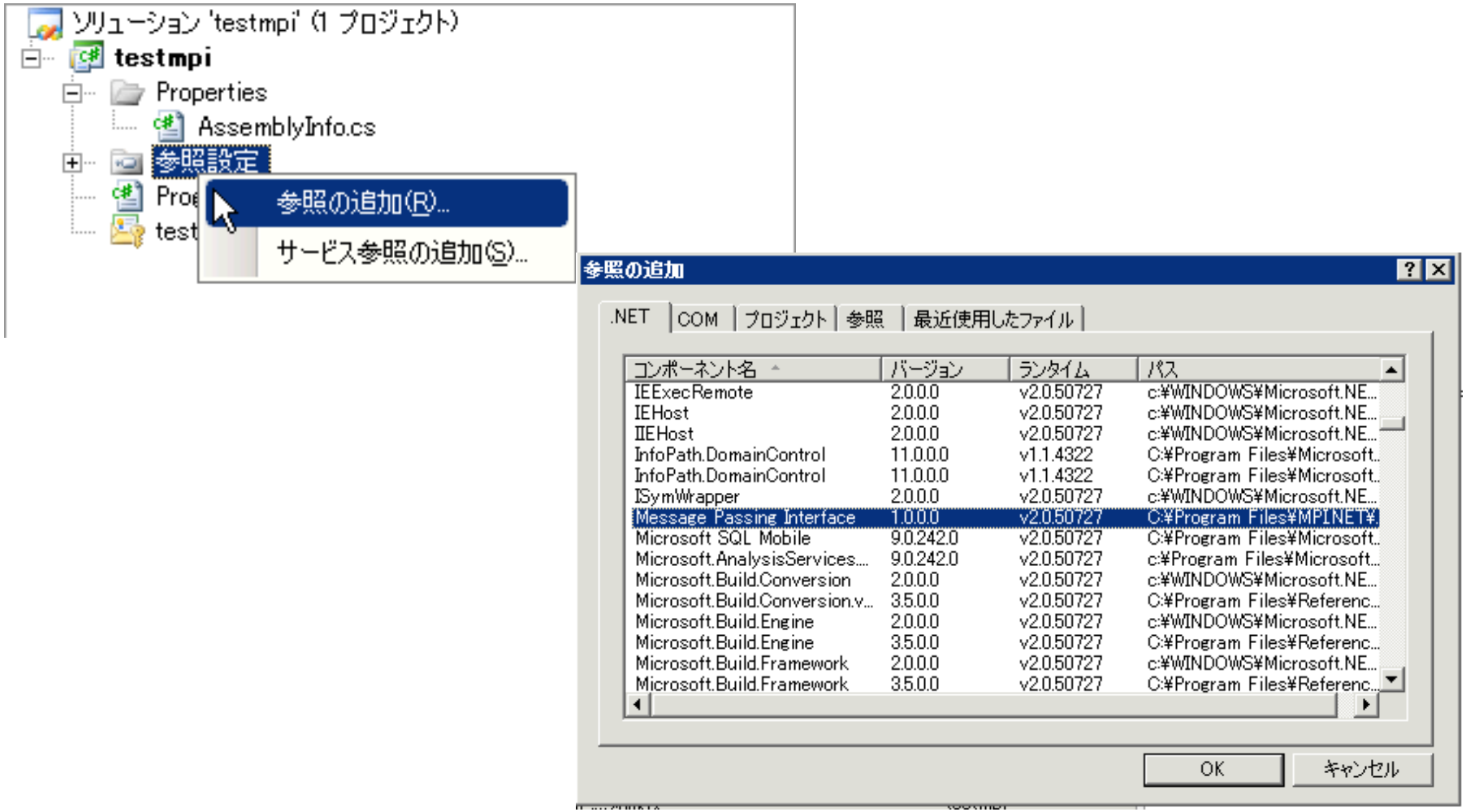
プログラム作成準備(1/2)

1. Visual Studio 2008を起動する
2. 「新規作成」->「プロジェクト」->
   「コンソールアプリケーション」を選択



プログラム作成準備(2/2)

3. ソリューションエクスプローラの参照設定を右クリック
4. 「参照の追加」で「Message Passing Interface」を選択



1対1通信のプログラム作成例(1/2)

HelloWorldという文字列をランク0がランク1に送る

```
using System;
using MPI;                          // MPI環境のための名前空間

class Helloworld
{
    static void Main(string[] args)
    {   // MPIインスタンスの作成（環境の初期化）
        using (new MPI.Environment(ref args))
        {
            （MPIプログラムをここに書く）
        }
    }
}
```

1対1通信のプログラム作成例(2/2)

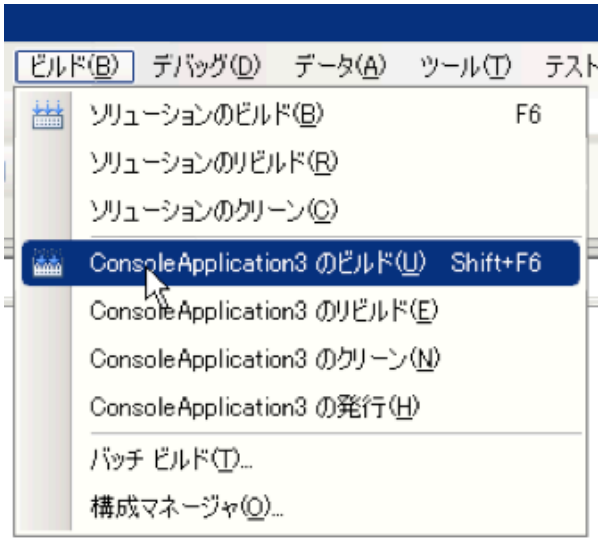
```
// コミュニケータの宣言（ランクの取得など）
Intracommunicator comm = Communicator.world;
int tag = 9; // 今回は適当な数字を代入

if (comm.Rank == 0)
{
    comm.Send(”HelloWorld”, 1, tag);  // ランク1に送信
}
else if(comm.Rank == 1)
{
    string msg = comm.Receive<string>(0, tag); // ランク0から受信
    Console.Write(msg);
}
```

コンパイル & 実行

■コンパイル方法

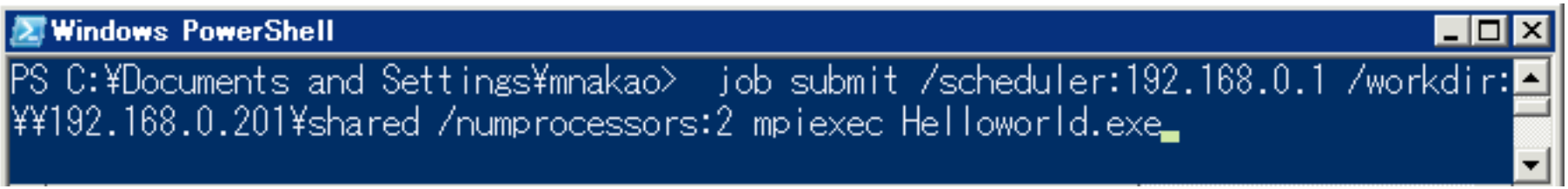
・Shift + F6

・「ビルド」 ->
　（プロジェクト名）のビルド



■実行方法（PowerShellを用いる方法）

> job submit /scheduler:(ジョブスケジューラ) /numcores:(プロセス数)
/workdir:(実行ファイルの場所) mpiexec (実行ファイル)



C言語とmpichの場合(1/2)

```
#include <stdio.h>
#include <string.h>
#include <mpi.h>     // using MPIに相当

int main(int argc, char *argv[])
{
  char msg[20];       // 送信用データ配列
  int rank;           // ランク
  int tag = 9;        // タグ

  MPI_Init(&argc, &argv);
  MPI_Comm_rank(MPI_COMM_WORLD, &rank);
```

new MPI.Environment(ref args)
に相当

C言語とmpichの場合(2/2)

```
if (rank == 0)
{
    sprintf(msg, “HelloWorld”);
    MPI_Send(msg, strlen(msg), MPI_CHAR, 1, tag, MPI_COMM_WORLD);
}
else if(rank == 1)
{
     MPI_Status status;
     MPI_Recv(msg, 20, MPI_CHAR, 0, tag, MPI_COMM_WORLD, &status);

    printf(“%s\n”, msg);
}
  MPI_Finalize();
  return 0;
}
```

引数が多い

MPI.NETとmpichの送信関数の比較

■ MPI.NET

comm.Send(value, dest, tag)

引数の数は3つ（インスタンスを含めると4つ）

■ mpich

```
MPI_Send(void *msg, int count, MPI_Datatype datatype,
                                    int dest, int tag, MPI_COMM comm)
```

・msg：送信データの先頭アドレス
・count：送信データの数
・MPI_Datatype：送信データの型
・dest：送信先のランク
・tag：識別タグ
・comm：コミュニケータ

引数の数は6つ

集合通信

例えばあるプロセスが複数のプロセスにデータを送りたい場合、

```
for(i=0; i<num; i++){
   comm.send(“HelloWorld”, i, tag);
}
```

この書き方だとコードが複雑化する。

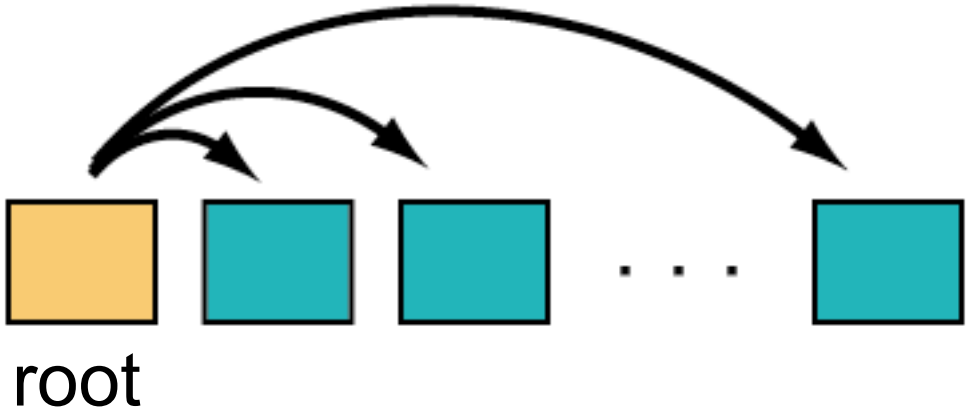
MPIではone-to-all, all-to-one, all-to-allの関数（集合通信）が定義されている。
集合通信用関数を用いた方がパフォーマンスも高い。
（例えば二分木で送信が可能）

集合通信の例：one to all

public void Broadcast<T>(ref T value, root)

1つのプロセスから全てのプロセスにデータを
送信するための関数

- ref T value：送信する値
- root：送信するランク

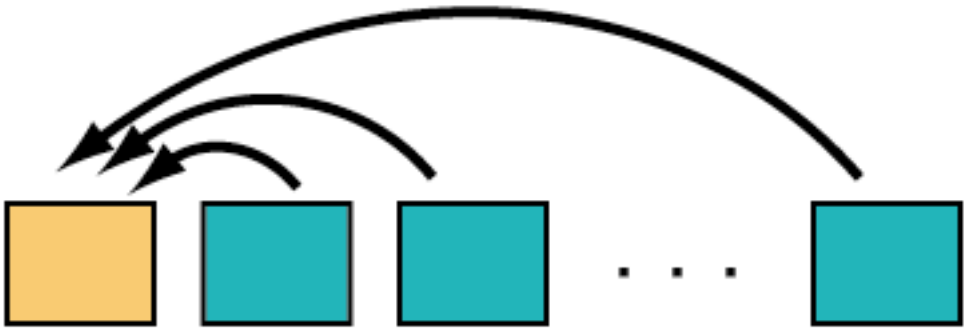


集合通信の例 : all to one

public T[] Gather<T>(value, root)

あるプロセスに全てのプロセスからデータを
受信するための関数

- value：集めるデータ
- root　：受信するランク
- 返り値：集めた値の配列



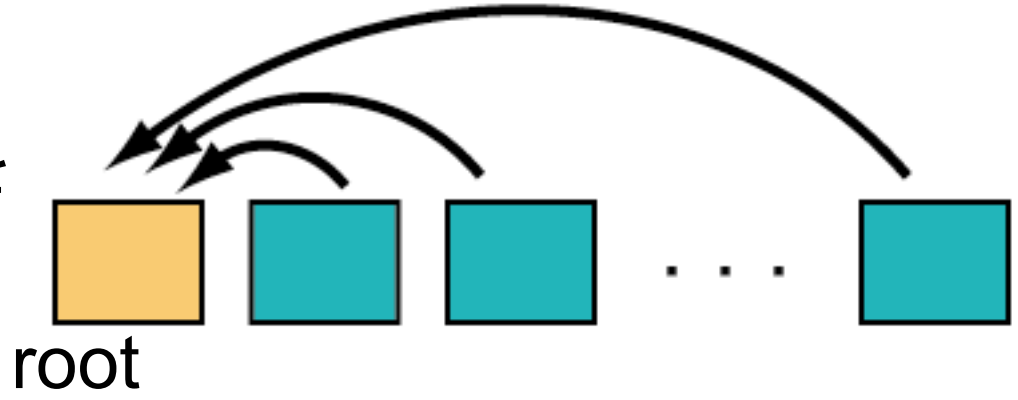
集合通信の例：Reduction(集約)

public T Reduce<T>(value, Operation, root)

あるプロセスに全てのプロセスからデータを
受信するための関数。受信時に算術演算を行える。
並列計算においてよく使う計算を提供している。

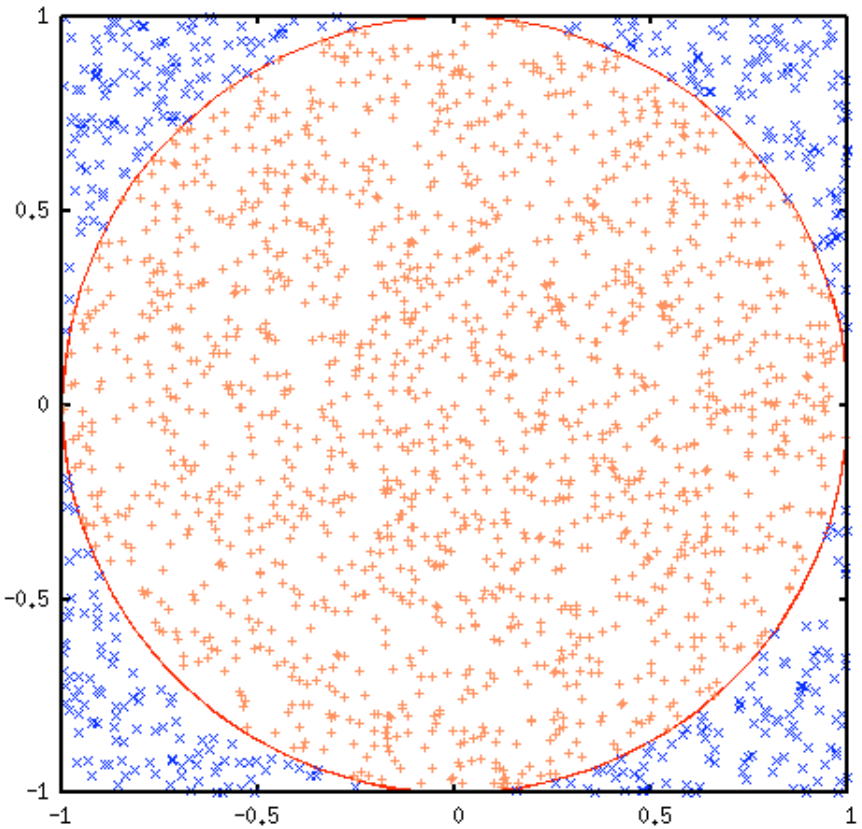
- value：送信する値
- Opration：算術演算（自分でも定義できる）
- root：受信するランク

受信するデータ群の
最大値、合計値などを
自動的に計算



モンテカルロ法によるπ計算(1/3)

-1から1の乱数を多く発生させる.



$\pi = 4 \times$ 円内部の点の数 ÷ 全ての点の数

になる。

点の数が多いほど、
精確な値になる。

モンテカルロ法によるπ計算(2/3)

■モンテカルロ法の並列化

1. 複数のプロセスで乱数を発生させる。
2. 各プロセスで円の内部の点を数え、
   その値をランク0にReduceを使って送信
3. ランク0はReduceを使って、円の内部の点の総計を
   受信する
4. ランク0はπの値を計算する

プログラムはhttp://www.osl.iu.edu/research/mpi.net/を参考

モンテカルロ法によるπ計算(3/3)

```
Random random = new Random(comm.Rank); // 乱数の種
int num = 10000; // 発生させる乱数の数
int count = 0;  // 円の内部に発生する点の数

for (int i = 0; i < num; ++i)
{
    double x = (random.NextDouble() - 0.5) * 2;
    double y = (random.NextDouble() - 0.5) * 2;
    if (x * x + y * y <= 1.0)
        ++ count;
}

int total = comm.Reduce(count, Operation<int>.Add, 0);
if (comm.Rank == 0)
      Pi =  4*(double) total/(comm.Size*(double)num));
```

まとめ

■C#とMPI.NETを用いた並列プログラミングについての
概要の説明

■MPI.NETの基本的な用語の説明と通信関数についての説明

■実際のプログラミング例の説明

MPI.NETとC#を用いると、簡易に並列プログラムを
開発できます。

参考文献など

- MPI.NET(本家)
  http://www.osl.iu.edu/research/mpi.net/
  インストール方法、プログラム例が豊富

- MPI.NETの関数のリファレンス
  http://www.osl.iu.edu/research/mpi.net/documentation/reference/current/Index.html