Asignación de Prácticas Número 10

Programación Concurrente y de Tiempo Real

Departamento de Ingeniería Informática
Universidad de Cádiz

PCTR, 2022

Contenido

Objetivos de la práctica

- Efectuar una aproximación elemental a este API de paso de mensajes.
- Aprender a comunicar procesos con primitivas send/receive bloqueantes, y con broadcast/reduce.
- Aplicar lo anterior a algunos problemas sencillos que ilustren los aspectos básicos del modelo.

Procesos y MPJ-Express

- MPJ-Express trabaja con procesos en lugar de hebras.
- No existe memoria común.
- La comunicación se efectúa intercambiando mensajes que transportan los datos que interesan.
- La programación de soluciones cooperativas entre procesos es probablemente más compleja.
- Se aplica en la resolución paralela de problemas asociados a computación científica, normalmente en ambientes *cluster*.

Creando Procesos con MPJ-Express

- Programamos la estructura del proceso necesaria.
- Pedimos a `mpjrun` (es un fichero de procesamiento por lotes) la creación de tantas instancias del proceso como se quiera...
- ... pero esto solo nos da mútiples procesos que hacen exactamente lo mismo.
- Solución: hacer que procesos diferentes ejecuten regiones de código distintas, utilizando para el identificador de cada uno de los procesos, que es diferente.
- Es muy parecido a la concurrencia creando procesos en lenguaje C con llamadas al sistema `fork()`.

Ejemplo Inútil

```
1  import mpi.*;
2  public class ejInutil{
3
4    public static void main(String args[]) throws Exception {
5      MPI.Init(args);
6      int me = MPI.COMM_WORLD.Rank();
7      int size = MPI.COMM_WORLD.Size();
8      System.out.println("Soy el proceso <"+me+">");
9      MPI.Finalize();
10   }
11 }
```

Ejemplo Algo Más Útil

```
1  import mpi.*;
2  public class ejMasUtil{
3
4    public static void main(String args[]) throws Exception {
5      MPI.Init(args);
6      int me = MPI.COMM_WORLD.Rank();
7      int size = MPI.COMM_WORLD.Size();
8      if(me==0){
9        int a,b;
10       a=10; b=5;
11       int c=a+b;
12       System.out.println("Soy el proceso <"+me+"> y a+b es
             igual a "+c);
13     }
14     else{System.out.println("Soy el proceso <"+me+"> y a mi no
           me gusta sumar");}
15
16     MPI.Finalize();
17   }
18 }
```

Transfiriendo un Array 1D con Send-Recv I

```
1  //transfiere un array de enteros del emisor al recpetor
2  //COMPILACION:javac -cp .;%MPJ_HOME%/lib/mpj.jar
       ArrayPuntoAPunto.java
3  //EJECUCION: mpjrun.bat -np 2 ArrayPuntoAPunto
4
5  import mpi.*;
6  import java.util.Arrays;
7  public class ArrayPuntoAPunto {
8
9  public static void main(String args[]) throws Exception {
10  MPI.Init(args);
11  int rank = MPI.COMM_WORLD.Rank();
12  int size = MPI.COMM_WORLD.Size();
13  int emisor = 0; int receptor = 1;
14  int tag = 100; int unitSize = 10;
15
16  if(rank==emisor){ //codigo del emisor
17    int bufer[] = new int[10];
18    for(int i=0; i<bufer.length; i++)bufer[i] = i;
19    MPI.COMM_WORLD.Send(bufer, 0, unitSize, MPI.INT, receptor,
          tag);
20  } else{ //codigo del receptor
```

Transfiriendo un Array 1D con Send-Recv II

```
21      int revbufer[] = new int[10];
22      MPI.COMM_WORLD.Recv(revbufer, 0, unitSize, MPI.INT, emisor,
            tag);
23      System.out.println("Recibido: "+Arrays.toString(revbufer));
24      }
25
26    MPI.Finalize();
27    }
28  }
```

Computando un Producto por Escalar de Ida y Vuelta con Send-Recv I

```
1  //transfiere un array de enteros del emisor al receptor, se
       hace un producto escalar, y se devuelve
2  //COMPILACION:javac -cp .;%MPJ_HOME%/lib/mpj.jar pEscalar.java
3  //EJECUCION: mpjrun.bat -np 2 pEscalar
4  import mpi.*;
5  import java.util.Arrays;
6  public class pEscalar {
7
8  public static void main(String args[]) throws Exception {
9   MPI.Init(args);
10  int rank = MPI.COMM_WORLD.Rank();
11  int size = MPI.COMM_WORLD.Size();
12  int emisor = 0; int receptor = 1;
13  int tag = 100; int unitSize = 10;
14
15  if(rank==emisor){ //codigo del emsior
16    int bufer[] = new int[10];
17    for(int i=0; i<bufer.length; i++)bufer[i] = i;
18    MPI.COMM_WORLD.Send(bufer, 0, unitSize, MPI.INT, receptor,
          tag);
```

Computando un Producto por Escalar de Ida y Vuelta con Send-Recv II

```
19              MPI.COMM_WORLD.Recv(bufer, 0, unitSize, MPI.INT,
                    receptor, tag);
20              System.out.println("Emisor ha recibido:
                    "+Arrays.toString(bufer));
21      } else{ //codigo del receptor
22        int revbufer[] = new int[10];
23        MPI.COMM_WORLD.Recv(revbufer, 0, unitSize, MPI.INT, emisor,
              tag);
24        System.out.println("Receptor ha recibido:
              "+Arrays.toString(revbufer));
25              int k =10;
26              for(int i=0; i<revbufer.length; i++)revbufer[i] =
                    revbufer[i]*k;
27              MPI.COMM_WORLD.Send(revbufer, 0, unitSize, MPI.INT,
                    emisor, tag);
28          }
29       MPI.Finalize();
30       }
31      }
```

Calculadora Distribuida Con Send-Recv I

```
1  //calculadora distribuida cutre...
2  //COMPILAR: javac -cp .;%MPJ_HOME%/lib/mpj.jar
       calculadoraDistribuida.java
3  //EJECUTAR: mpjrun.bat -np 5 calculadoraDistribuida
4
5  import mpi.*;
6  public class calculadoraDistribuida{
7
8  public static void main(String args[]) throws Exception {
9   MPI.Init(args);
10  int rank = MPI.COMM_WORLD.Rank();
11  int size = MPI.COMM_WORLD.Size();
12  int emisor = 0; int tag = 100; int unitSize = 1;
13
14  if(rank==emisor){
15    int bufer[] = new int[2];
16    bufer[0] = 4;
17    bufer[1] = 3;
18          for(int i=1; i<size; i++){
19      MPI.COMM_WORLD.Send(bufer, 0, unitSize, MPI.INT, i, tag+i);
20            MPI.COMM_WORLD.Send(bufer, 1, unitSize, MPI.INT, i,
                  tag+i);
```

Calculadora Distribuida Con Send-Recv II

```
21      }
22    } else if(rank==1){
23      int res;
24            int revbufer[] = new int[2];
25      MPI.COMM_WORLD.Recv(revbufer, 0, unitSize, MPI.INT, emisor,
            tag+rank);
26      MPI.COMM_WORLD.Recv(revbufer, 1, unitSize, MPI.INT, emisor,
            tag+rank);
27            res = revbufer[0]+revbufer[1];
28      System.out.println("Suma: "+res);
29      }else if(rank==2){
30      int res;
31            int revbufer[] = new int[2];
32      MPI.COMM_WORLD.Recv(revbufer, 0, unitSize, MPI.INT, emisor,
            tag+rank);
33      MPI.COMM_WORLD.Recv(revbufer, 1, unitSize, MPI.INT, emisor,
            tag+rank);
34            res = revbufer[0]-revbufer[1];
35      System.out.println("Resta: "+res);
36          }else if(rank==3){
37      int res;
38            int revbufer[] = new int[2];
```

Calculadora Distribuida Con Send-Recv III

```
39      MPI.COMM_WORLD.Recv(revbufer, 0, unitSize, MPI.INT, emisor,
            tag+rank);
40      MPI.COMM_WORLD.Recv(revbufer, 1, unitSize, MPI.INT, emisor,
            tag+rank);
41            res = revbufer[0]*revbufer[1];
42      System.out.println("Producto: "+res);
43          }else if(rank==4){
44      float res;
45            int revbufer[] = new int[2];
46      MPI.COMM_WORLD.Recv(revbufer, 0, unitSize, MPI.INT, emisor,
            tag+rank);
47      MPI.COMM_WORLD.Recv(revbufer, 1, unitSize, MPI.INT, emisor,
            tag+rank);
48            if(revbufer[1]!=0){res =
                  revbufer[0]/(float)revbufer[1];
                  System.out.println("Cociente: "+res);}
49            else System.out.println("No se puede dividir por
                  cero...");
50          }
51    MPI.Finalize();
52    }}
```

Calculadora Distribuida Con Send-Recv Mejorada I

```
1  //calculadora distribuida algo mejor...
2
3  import mpi.*;
4  public class calculadoraDistribuida2 {
5
6  public static void main(String args[]) throws Exception {
7   MPI.Init(args);
8   int rank = MPI.COMM_WORLD.Rank();
9   int size = MPI.COMM_WORLD.Size();
10  int emisor = 0;
11  int tag = 100; int unitSize = 2;
12
13  if(rank==emisor){
14    int bufer[] = new int[2];
15    bufer[0] = 4;
16    bufer[1] = 3;
17          for(int i=1; i<size; i++){
18      MPI.COMM_WORLD.Send(bufer, 0, unitSize, MPI.INT, i, tag+i);
19    }
20  } else if(rank==1){
21    int res;
22          int revbufer[] = new int[2];
```

Calculadora Distribuida Con Send-Recv Mejorada II

```
23      MPI.COMM_WORLD.Recv(revbufer, 0, unitSize, MPI.INT, emisor,
            tag+rank);
24            res = revbufer[0]+revbufer[1];
25      System.out.println("Suma: "+res);
26      }else if(rank==2){
27      int res;
28            int revbufer[] = new int[2];
29      MPI.COMM_WORLD.Recv(revbufer, 0, unitSize, MPI.INT, emisor,
            tag+rank);
30            res = revbufer[0]-revbufer[1];
31      System.out.println("Resta: "+res);
32          }else if(rank==3){
33      int res;
34            int revbufer[] = new int[2];
35      MPI.COMM_WORLD.Recv(revbufer, 0, unitSize, MPI.INT, emisor,
            tag+rank);
36            res = revbufer[0]*revbufer[1];
37      System.out.println("Producto: "+res);
38          }else if(rank==4){
39      float res;
40            int revbufer[] = new int[2];
```

Calculadora Distribuida Con Send-Recv Mejorada III

```
41      MPI.COMM_WORLD.Recv(revbufer, 0, unitSize, MPI.INT, emisor,
            tag+rank);
42            if(revbufer[1]!=0){res =
                  revbufer[0]/(float)revbufer[1];
                  System.out.println("Cociente: "+res);}
43            else System.out.println("No se puede dividir por
                  cero...");
44          }
45    MPI.Finalize();
46
47    }
48  }
```

Calculadora Distribuida Con Bcast I

```
1  //calculadora distribuida con broadcast...
2  import mpi.*;
3  public class calculadoraDistribuida3 {
4
5  public static void main(String args[]) throws Exception {
6   MPI.Init(args);
7   int rank = MPI.COMM_WORLD.Rank();
8   int size = MPI.COMM_WORLD.Size();
9   int emisor = 0;
10  int tag = 100;
11  int unitSize = 2;
12
13    int bufer[] = new int[2];
14    if(rank==0){
15      bufer[0] = 4;
16      bufer[1] = 3;
17    }
18
19 MPI.COMM_WORLD.Bcast(bufer, 0, unitSize, MPI.INT, 0);
20
21    if(rank==1){
22      int res;
```

Calculadora Distribuida Con Bcast II

```
23        res = bufer[0]+bufer[1];
24        System.out.println("Suma: "+res);
25       }else if(rank==2){
26        int res;
27              res = bufer[0]-bufer[1];
28        System.out.println("Resta: "+res);
29            }else if(rank==3){
30              int res;
31                    res = bufer[0]*bufer[1];
32              System.out.println("Producto: "+res);
33                  }else if(rank==4){
34                    float res;
35                          if(bufer[1]!=0){
36                            res = bufer[0]/(float)bufer[1];
37                            System.out.println("Cociente: "+res);}
38                          else System.out.println("No se puede
                                dividir por cero...");
39            }
40     MPI.Finalize();
41     }
42   }
```

¿Qué Hago Ahora?

- Repase en el API de MPJ-Express el uso de los métodos Send, Recv, Bcast y Reduce.
- Resuelva el primer ejercicio de la asignación utilizando Send y Recv.
- Resuelva el segundo ejercicio de la asignación utilizando Bcast.
- Resuelva el tercer ejercicio de la asignación utilizando Bcast-Reduce.