PRACTICA 4. ARQUITECTURA DE COMPUTADORES

JOSE IGNACIO MORENO MONTESINOS Y DIEGO GONZALEZ SANZ

INDICE

Ejercicio 1:

En este ejercicio se nos pide realizar un ejemplo de las primitivas de MPI (MPI_Reduce, MPI_Allgather y MPI_Alltoall) y utilizarlo para explicar el funcionamiento de cada una de ellas.

Primero lo que hacemos es MPI_Reduce para realizar la operación de la suma utilizando el array que creamos y guardándolo en recep1. Luego con el MPI_AlltoAll, lo que hacemos es dividir los valores que utilizamos para cada proceso. En términos más teóricos, lo que hace es que el búfer de envío en cada proceso se divide y luego cada columna de fragmentos se recopila mediante el proceso respectivo, cuyo rango coincide con el número de la columna de fragmentos.

Por último, utilizamos el MPI_AllGather, que lo que hace es coger un conjunto de datos distribuidos ente los distintos nodos del comunicador, y los datos se recogen en todos los procesos.

Para cada procedimiento hemos utilizado bucles for tanto para los procesos como para los datos, como se muestra a continuación:

```
alumnos@sistred08-ThinkStation-P320:~/Escritorio/Practica4AC$ mpirun -np 3 ./ejecutable1
Suma: 6  6  6

Values collected on process (ALLTOALL) 0: 1,
Values collected on process (ALLTOALL) 1: 2,
Values collected on process (ALLTOALL) 2: 3,

Values collected on process (ALLGATHER) 0: 1,2,3,
Values collected on process (ALLGATHER) 1: 1,2,3,
Values collected on process (ALLGATHER) 2: 1,2,3,
alumnos@sistred08-ThinkStation-P320:~/Escritorio/Practica4AC$
```

Ejercicio 2:

En este ejercicio se nos pide que construyamos una matriz y que se haga la transposición de la misma.

Creamos dos variables, una de envío y otra de recepción. Ambas son arrays. Después hacemos un if, que comprueba que por consola se introduce la cantidad de 4 procesos, ya que son los que se necesitan para poder construir la matriz. En caso de introducir cualquier otro número de procesos, nos dirá que deben de ser 4 los procesos necesarios.

Luego tenemos un for que lo que hace es introducir los valores de la matriz dentro del array de envío. Sumamos una posición en i y luego lo sumamos el rank y multiplicamos por el size, para que, en cada posición del array, salgan los valores ordenados y con el mismo valor que la posición. Dentro del for hay una linea comentada, que hace lo mismo que antes, pero añadiéndole un random y una función llamada fabs(), que lo que hace es convertir los números negativos en números positivos. Es necesario, ya que haciendo el random salen números negativos.

Luego mostramos la matriz de envío con todos sus valores, y cada proceso muestra una fila de la matriz. Después, hacemos el MPI_AlltoAll, en el que le pasamos el buffer de envío la variable envío y como buffer recibidor la variable de recepción. Esta función lo que hace para cada proceso, es que te muestra un valor para cada proceso, y en este caso al tratarse de una matriz, lo hace convirtiendo las filas en columnas.

Luego, por último, imprimimos la matriz de recepción, y queda como se muestra a continuación:

```
nacho@nacho-ubuntu:~/Escritorio/P4AC$ mpicc ejercicio2.c -o ejecutable2
nacho@nacho-ubuntu:~/Escritorio/P4AC$ mpirun -np 4 ./ejecutable2
5 6 7 8
9 10 11 12
13 14 15 16
1 2 3 4
Proceso: 3 : 4 8 12 16
Proceso: 0 : 1 5 9 13
Proceso: 2 : 3 7 11 15
Proceso: 1 : 2 6 10 14
nacho@nacho-ubuntu:~/Escritorio/P4AC$
```

Ejercicio 3:

Para que este programa compile debemos poner esta linea ya que si no nos saldrá un error

mpicc ejercicio3.c -o ejecutable3 –lm. Tenemos que poner –lm porque es una opción para enlazar con la biblioteca matemática (libm).

Primero declaramos la variable size y rank para almacenar en size el tamaño de procesos que tenemos que hacer y rank para saber el proceso donde estamos. Después hacemos un variable para almacenar números aleatorios entre 0 y 1, y después una variable resultada que será la suma de todo el numero binaro y el multiplicador que es el resultado de la posición del número (más tarde explico con detalle lo que hace).

Utilizamos el MPI_Allgather el primer parámetro es que envía un número, el segundo parámetro es cuanto es el tamaño que envías, el cuarto parámetro es donde lo recibe y solo recibe uno. Después hacemos una condición if que es si el numero aleatorio es igual a 1 nos metemos en esa condición y si el numero aleatorio es 0 el multiplicador es 0. Dentro de que el numero aleatorio es 1 tenemos que hacer otra condición que es si el rank que es el proceso donde estamos es distinto a él (tamaño de todos los procesos – 1) porque nosotros empezamos en el proceso 0 y el último proceso es size – 1, hacemos un elevado de 2^(size – rank -1), esto debemos hacerlo así ya que como tenemos un ejemplo en el documento el último proceso empieza valiendo 0 o 1, el penúltimo proceso puede valer o 0 o 2 y así continuamente hasta el proceso 0. Después utilizamos MPI_REDUCE que el primer parámetro es la información de donde la recibo y el segundo parámetro donde voy a enviar esa información y hacemos una suma.

Para finalizar hacemos que cuando sea el proceso 0 nos muestre el número en binario y además el resultado de ese número en binario.

Solución con 2 procesos:

```
diego@diego-VivoBook-ASUSLaptop-X415DA-M415DA:~/Arquitectura de computadores$ mpirun -np 2 ./ejecutable3
Proceso 0 , dato: 0
Numeros binarios: 00
Soy el proceso 0 y el numero decimal es: 0
Proceso 1 , dato: 0
```

Solución con 6 procesos:

```
diego@diego-VivoBook-ASUSLaptop-X415DA-M415DA:~/Arquitectura de computadores$ mpirun -np 6 ./ejecutable3
Proceso 0 , dato: 0
Proceso 1 , dato: 0
Proceso 2 , dato: 1
Proceso 3 , dato: 1
Proceso 4 , dato: 1
Proceso 5 , dato: 1
Numeros binarios: 001111
Soy el proceso 0 y el numero decimal es: 15
```

Solución con 10 procesos:

```
diego@diego-VivoBook-ASUSLaptop-X415DA-M415DA:~/Arquitectura de computadores$ mpirun -np 10 ./ejecutable3
Proceso 2 , dato: 1
Proceso 3 , dato: 0
Proceso 4 , dato: 0
Proceso 6 , dato: 0
Proceso 1 , dato: 0
Proceso 9 , dato: 0
Proceso 5 , dato: 1
Proceso 7 , dato: 1
Proceso 0 , dato: 0
Proceso 8 , dato: 1
Numeros binarios: 0010010110
Soy el proceso 0 y el numero decimal es: 150
```