MEMORIA P3

Zdravko Dimitrov Arnaudov

Índice

1.

Tenemos que usar el programa de los trapezoides y modificarlo para que el proceso raíz tome los datos de entrada y los transmita al resto con un Broadcast, comprobando antes y después del envío que se ha realizado correctamente.

Copiamos el programa de trapezoides proporcionado en la práctica anterior. La modificación que tenemos que hacer, consiste en usar la función MPI_Bcast para retransmitir las métricas que el proceso raíz lee por teclado.

```
//antes de la llamada
  printf ("Soy el proceso: %d. Tengo A: %d, B: %d y N: %d\n", yo, a, b,
n);

  MPI_Bcast(&a, 1, MPI_FLOAT, 0, MPI_COMM_WORLD);
  MPI_Bcast(&b, 1, MPI_FLOAT, 0, MPI_COMM_WORLD);
  MPI_Bcast(&n, 1, MPI_INT, 0, MPI_COMM_WORLD);

  //despues de la llamada
  printf ("Soy el proceso: %d. Tengo A: %d, B: %d y N: %d\n", yo, a, b,
n);
```

La salida que obtenemos al ejecutar el programa es la siguiente:

```
mpirun -np 4 Ejer1
Soy el proceso: 1. Tengo A: 0.000000, B: 10.000000 y N: 1000
Soy el proceso: 3. Tengo A: 0.000000, B: 10.000000 y N: 1000
Soy el proceso: 0. Tengo A: 0.000000, B: 10.000000 y N: 1000
Mete a , b, n
Soy el proceso: 2. Tengo A: 0.000000, B: 10.000000 y N: 1000
0
10000000000
100
Soy el proceso: 0. Tengo A: 0.000000, B: 10000000000.000000 y N: 100
Soy 0 inicio y fin son 0.000000 - 2500000000.000000 integral ini es
3124999787749834752.000000 h 100000000.000000
Soy el proceso: 2. Tengo A: 0.000000, B: 10000000000.000000 y N: 100
```

```
Soy 2 inicio y fin son 5000000000.000000 - 7500000256.000000 integral
ini es 15624994815580569600.000000 h 100000000.000000
Soy el proceso: 1. Tengo A: 0.000000, B: 10000000000.000000 y N: 100
Soy 1 inicio y fin son 2500000000.000000 - 5000000000.000000 integral
ini es 9374997988859969536.000000 h 100000000.000000
Soy el proceso: 3. Tengo A: 0.000000, B: 10000000000.000000 y N: 100
Soy 3 inicio y fin son 7500000256.000000 - 10000000000.000000 integral
ini es 21874992741812797440.000000 h 100000000.000000
Con 100 trapezoides la estimacion es de 49999983409857822720.000000
```

Por último, responderemos a las preguntas que vienen indicadas en el ejercicio:

a) No es necesario poner una barrera MPI después de llamada Broadcast porque ya es bloqueante. Es decir, se continúa con el código cuando el envío ha sido completado en todos los procesos.

b) Si usamos scanf en todos los procesos, vamos a tener que repetir las operaciones tantas veces como procesos corran el programa.

2.

Veamos en detalle el planteamiento del ejercicio 2.

El objetivo es multiplicar una matriz por un vector y paralelizar la operación. A cada proceso, se le asignarán uno o varios paneles de la matriz, así como filas del vector si existen más columnas que procesos.

En cuanto al flujo del programa, primero deben conocerse las dimensiones de la matriz a multiplicar. Para esta tarea, intervendrá el proceso raíz 0. Cuando el máster haya transmitido las métricas al resto de procesos, podrán continuar todos trabajando.
Después, cada proceso determina la porción de trabajo que debe hacer. En particular, se adquiere la fracción de la matriz que le corresponde y se inicializa con los valores adecuados, al igual que con el vector.

Cuando los procesos han inicializado su parte del vector, lo mandan al resto de procesos y reciben de ellos, para que cada uno componga el vector entero y pueda multiplicar la matriz.

Llegado a este punto, toma lugar la multiplicación en sí, donde los procesos almacenarán los resultados en un vector de solución propio. Finalmente, imprimimos el vector solución completo por pantalla.

Esta es la salida que se obtiene con una matriz 8x8:

```
mpirun -np 4  Ejer2
Escribe las dimensiones de la matriz (MxN):
8
8
PROC3-VEC_RES[6]: 1820.00
PROC3-VEC_RES[7]: 2100.00
PROC0-VEC_RES[0]: 140.00
PROC1-VEC_RES[2]: 700.00
PROC1-VEC_RES[3]: 980.00
PROC2-VEC_RES[4]: 1260.00
PROC2-VEC_RES[5]: 1540.00
PROC0-VEC_RES[1]: 420.00
```

3.

Se proporciona un programa secuencial que escribe 100 dobles en un fichero. Se debe crear un programa MPI, donde el proceso raíz lea del fichero los 100 valores y los reparta a los N procesos para que los impriman por pantalla.

Emplearemos la llamada Scatter para que root reparta, por partes iguales, los números a todos los procesos. Por último, los procesos imprimen los dobles que tienen asignados. Este es el resultado que obtenemos:

```
mpirun -np 4 Ejer3 file
Soy el proceso raiz, dobles leidos.
Soy 0:  0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
21 22 23 24
Soy 1: 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45
46 47 48 49
Soy 2: 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70
71 72 73 74
Soy 3: 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95
96 97 98 99
```

4.

a)

Para este ejercicio, reutilizaremos el código del apartado 1, en el que enviamos las métricas a los demás procesos con Broadcast. La diferencia es que ahora, cuando todos hayan hecho sus cálculos, el proceso raíz hará una reducción de tipo suma con todos los valores y no tiene que hacerlo él manualmente.

El código que hemos añadido es el siguiente:

```
//En este momento cada proceso tiene que haber calculado su parte, es
hora de reducirlo
  MPI_Reduce (&integral, &total, 1, MPI_FLOAT, MPI_SUM, 0,
MPI_COMM_WORLD);

  if (yo ==0){
     printf("Con %d trapezoides la estimacion es de %f\n",n,total);
  }
```

El resultado es este:

```
mpirun -np 4 Ejer4_A
Soy el proceso: 3. Tengo A: 0.000000, B: 10.000000 y N: 1000
Soy el proceso: 0. Tengo A: 0.000000, B: 10.000000 y N: 1000
Mete a , b, n
Soy el proceso: 1. Tengo A: 0.000000, B: 10.000000 y N: 1000
Soy el proceso: 2. Tengo A: 0.000000, B: 10.000000 y N: 1000
0
1000000
1000
Soy el proceso: 0. Tengo A: 0.000000, B: 1000000.000000 y N: 1000
Soy el proceso: 3. Tengo A: 0.000000, B: 1000000.000000 y N: 1000
Soy 3 inicio y fin son 750000.000000 - 1000000.000000 integral ini es
218749665280.000000 h 1000.000000
Soy el proceso: 1. Tengo A: 0.000000, B: 1000000.000000 y N: 1000
Soy 1 inicio y fin son 250000.000000 - 500000.000000 integral ini es
93750001664.000000 h 1000.000000
Soy el proceso: 2. Tengo A: 0.000000, B: 1000000.000000 y N: 1000
Soy 2 inicio y fin son 500000.000000 - 750000.000000 integral ini es
156249882624.000000 h 1000.000000
Soy 0 inicio y fin son 0.000000 - 250000.000000 integral ini es
31249999872.000000 h 1000.000000
Con 1000 trapezoides la estimacion es de 499999539200.000000
```

Hemos verificado por nuestra cuenta que la estimación es correcta, comparándola con el código del primer apartado y estos mismos parámetros.

b)

Debemos realizar el producto escalar de dos vectores de dimensión N, siendo N > P. En este caso, N no es necesariamente múltiplo del número de procesos. Si N no acaba siendo múltiplo de P, entonces el último proceso (p-1) se encargará de computar la parte restante.

Cuando los procesos terminen de ejecutar su porción, el proceso raíz reduce e imprime el resultado por pantalla. Esto es lo que se obtiene al usar N=23 con 4 procesos.

```
mpirun -np 4 Ejer4_B
P0: Filas 0 - 5
P3: Filas 15 - 23
```

```
P2: Filas 10 - 15
P1: Filas 5 – 10

Soy P0, suma_parcial : 30.00
Soy P2, suma_parcial : 730.00
Soy P1, suma_parcial : 255.00
Soy P3, suma_parcial : 2780.00
Resultado: 3795.00
```

c)

Si todos los procesos quisieran el mismo resultado, entonces usaríamos la llamada MPI_Allreduce.

d)

Podemos usar de referencia el código que viene proporcionado en los apuntes. Realmente, se trata de una reducción de tipo producto donde cada proceso aporta a la operación su rango +1 y el resultado se recoge en el proceso raíz indicado.

```
root=7;
source=rank+1;
MPI_Reduce(&source,&result,1,MPI_INT,
MPI_PROD,root,MPI_COMM_WORLD);
```

5.

a)

Sustituiremos la función de los trapezoides por **`sqrt(1 - pow(x, 2))`** de -1 a 1 para estimar Pi. Nos basaremos en la versión secuencial del programa de trapezoides.

```
./Ejer5_A
Mete a ,b, n
-1
1
1000
Con 1000 trapezoides la estimacion es de 1.570750
```

Como vemos, se aproxima bastante a Pi/2.

b)

Ahora, haremos la aproximación, pero usando el método de Montecarlo. Hay que generar puntos (x,y) aleatorios de 0 a 1 y ver cuántos caen dentro del primer cuadrante en una circunferencia de radio 1. Para medir la distancia, usaremos Pitágoras. El proceso máster, enviará un número N de puntos aleatorios que los demás procesos deben generar y al obtener la cuenta, enviarlo a root para que calcule el resultado final.

```
mpirun -np 4 Ejer5_B
Introduce dimensión de N aleatorios para cada proceso:
10000000
P0: puntos dentro: 7859199
P2: puntos dentro: 7861938
P1: puntos dentro: 7861482
P3: puntos dentro: 7864786
PI: 3.1447
```

6.

a)

Este apartado fue ya resuelto en el ejercicio 2, al haber empelado Allgather.

b)

Existen cuatro vagones con 4 pasajeros, uno para cada proceso. Cada vagón está representado por un array de elementos con número y nombre. Dadas unas circunstancias, los pasajeros ya no viajarán en un vagón en función de su clase, sino que cada clase irá separada en un vagón distinto.

La llamada que vamos a usar para resolver esta nueva ordenación es Alltoall. También, tener en cuenta que para transmitir un array de estructuras es necesario convertirlo todo a bytes y luego enviarlo.

```
mpirun -np 4 Ejer6_B
Soy 2: 1 2 3 4
Soy 3: 1 2 3 4
Soy 1: 1 2 3 4
Soy 0: 1 2 3 4
Soy 1: 2 2 2 2
Soy 3: 4 4 4 4
Soy 2: 3 3 3 3
Soy 0: 1 1 1 1
```