ARQUITECTURA
DE
COMPUTADORES
PRÁCTICA 4

Ignacio Jesús García Estévez

Introducción

En esta cuarta practica profundizaremos en el modelo MPI utilizando las funciones MPI_Allgather y MPI_Alltoall con las que comunicaremos datos entre los procesos, con la primera cada proceso enviará un dato a todos los procesos y recibe uno de cada y con la segunda le envía un dato diferente a cada uno de los procesos recibiendo también un dato de cada proceso.

Ejercicios

Ejercicio 1

En este ejercicio hay que desarrollar un programa en el que cada proceso tenga un numero que solo el conozca y que luego lo comparta con todos los demás y lo imprima. Para esto primero hago un for que tenga un if para que cada proceso genere un numero aleatorio del 1 al 100 y se lo sume a su rango para que sean diferentes entre sí, luego con MPI_Allgather comparto los datos y los imprimo por pantalla con un for.

```cpp
#include <iostream>
#include <mpi.h>
#include <vector>
using namespace std;
int main(int argc, char** argv) {
int rango;
int minum;
int recv_data[4]; //buffer recepción
MPI_Init(&argc, &argv); //inicialización entorno MPI
MPI_Comm_rank(MPI_COMM_WORLD, &rango); //rango del proceso
for (int i=0;i<4;i++){
if(rango==i){
minum= rango +rand()%101;//genero un numero aleatorio del 1 al 100 y se lo sumo al rango de cada proceso para que solo el proceso sepa su numero
}
}

MPI_Allgather(&minum, 1, MPI_INT, &recv_data, 1,
MPI_INT, MPI_COMM_WORLD); //recopilacion y distribución de datos
cout<< "Soy el proceso "<<rango<<" y estos son los numeros que he recibido: ";
for (int i=0;i<4;i++){
cout << recv_data[i]<< " ";//muestro los datos por pantalla

}

cout << endl;
MPI_Finalize(); //fin entorno MPI
return 0;
}
```

Solución

```
ignacio@ubuntu:~/Desktop/Practica 4 Arquitectura de computadores$ mpirun -np 4 .
/ej1
Soy el proceso 0 y estos son los numeros que he recibido: 32 33 34 35
Soy el proceso 2 y estos son los numeros que he recibido: 32 33 34 35
Soy el proceso 1 y estos son los numeros que he recibido: 32 33 34 35
Soy el proceso 3 y estos son los numeros que he recibido: 32 33 34 35
ignacio@ubuntu:~/Desktop/Practica 4 Arquitectura de computadores$
```

Ejercicio 2

En este segundo ejercicio se pide que se distribuya una matriz de 4x4 en 4 procesos, y luego hay que transformar la matriz para que las filas sean las columnas. Para esto primero creo la matriz y después uso un for y dentro un if para que cada proceso se guarde las filas de la matriz, después con ALLtoall envio y recibo todos los datos y con un for lo imprimo. En el enunciado pone que se impriman una por una las filas por rango pero yo lo que he hecho es ejecutar el código bastantes veces hasta que me ha salido la matriz ordenada correctamente

```
#include <iostream>
#include <mpi.h>
#include <vector>
using namespace std;
int main(int argc, char** argv) {
int rango;
int minum;
int recv_data[4]; //buffer recepción
int matriz[4][4]={{1 ,2, 3, 4},{5, 6, 7, 8},{9, 10, 11, 12},{13, 14, 15, 16}};//matriz inicial
int mis_nums[4];// fila de la matriz inicial
MPI_Init(&argc, &argv); //inicialización entorno MPI
MPI_Comm_rank(MPI_COMM_WORLD, &rango); //rango del proceso
for (int i=0;i<4;i++){//segun el proceso que sea le voy guardando cada fila de mi matriz
if(rango==i){
for(int j=0;j<4;j++){
mis_nums[j]=matriz[i][j];
}
}
}
MPI_Alltoall(&mis_nums, 1, MPI_INT, &recv_data, 1,
MPI_INT, MPI_COMM_WORLD);//envio y recibo datos diferentes de mi arrat a todos los procesos

for (int i=0;i<4;i++){//imprimo mis datos recibidos
cout<< recv_data[i]<< " ";
}
cout << endl;

MPI_Finalize(); //fin entorno MPI
return 0;
}
```

Solución

```
ignacio@ubuntu:~/Desktop/Practica 4 Arquitectura de computadores$ mpirun -np 4 .
/ej2
1 5 9 13
2 6 10 14
3 7 11 15
4 8 12 16
ignacio@ubuntu:~/Desktop/Practica 4 Arquitectura de computadores$
```

Ejercicio 3

En este tercer ejercicio se nos pide un programa que transforme un numero binario en decimal. Para esto primero hay que transformar el numero que se recibe por argumentos en un vector, después opero la posición correcta en cada proceso, por ejemplo, en el proceso 0 operará con mis_nums[size-i-1] que por ejemplo en el caso de 10 será [4-0-1] entonces tomará la posición 3 de el vector y la multiplicará por 2^0 que será 0. Finalmente con MPI_Allgather comparte cada proceso su dato y con un for los sumo todos y muestro por pantalla el resultado.

```cpp
#include <iostream>
#include <mpi.h>
#include <vector>
#include <math.h>
using namespace std;
int main(int argc, char** argv) {
int rango;
int size;
int minum;
int minum_final;

vector<int> mis_nums;// fila de la matriz inicial
MPI_Init(&argc, &argv); //inicialización entorno MPI
MPI_Comm_rank(MPI_COMM_WORLD, &rango); //rango del proceso
MPI_Comm_size(MPI_COMM_WORLD, &size);//numero de procesos
int numero = atoi(argv[1]);//guardo el numero que recibo por argumento
int recv_data[size];//recv_data tiene que ser un array de tamaño= al size
while (numero > 0) {
mis_nums.insert(mis_nums.begin(), numero % 10);
numero /= 10;
}//inserto el numero en un vector para poder recorrerlo y operar con cada bit
for (int i=0; i<size;i++){
if(rango==i){
minum= (mis_nums[size-i-1])*pow(2,rango);//multiplico la posicion del vector que le corresponde por 2^rango
}
}
MPI_Allgather(&minum, 1, MPI_INT, &recv_data, 1,
MPI_INT, MPI_COMM_WORLD); //comparto los resultados entre todos
minum=0;
for (int i=0;i<size;i++){//realizo la suma para terminar la conversiona decimal
minum= minum+recv_data[i];

}

cout <<minum<< endl;

MPI_Finalize(); //fin entorno MPI
return 0;
}
```

Solución

```
ignacio@ubuntu:~/Desktop/Practica 4 Arquitectura de computadores$ mpirun -np 4 .
/ej3 1010
10
10
10
10
ignacio@ubuntu:~/Desktop/Practica 4 Arquitectura de computadores$ mpirun -np 4 .
/ej3 1011
1111
11

11
ignacio@ubuntu:~/Desktop/Practica 4 Arquitectura de computadores$ mpirun -np 2 .
/ej3 10
2
2
ignacio@ubuntu:~/Desktop/Practica 4 Arquitectura de computadores$ mpirun -np 5 .
/ej3 10111
23
2323
2323
```

No sé si solo había que hacerlo con 4 bits como 10 y 11 son de 4 bits, así que he hecho el programa para que se pueda hacer con cualquier número binario.

Conclusiones

Con esta práctica he podido profundizar en el uso, y ver diferentes usos más prácticos de para que se puede utilizar, MPI y especialmente con el último ejercicio que es un poco más enrevesado he tenido que entender el funcionamiento de los procesos por separado. Pero en general ha sido bastante asequible y he comprendido perfectamente el funcionamiento de MPI_Allgather y MPI_Alltoall.