ARQUITECTURA
DE
COMPUTADORES
PRÁCTICA 2

Ignacio Jesús García Estévez

Introducción

Esta es la segunda practica de la asignatura, en la que afianzaremos aún mas nuestros conocimientos de MPI, realizando un ejercicio y una variación del mismo más complejos y que nos permitan entender más profundamente este estándar.

El objetivo principal de esta práctica es aprender entender de manera práctica la comunicación entre procesos, a través de varias nuevas funciones como MPI_ISend o MPI_IProbe o MPI_Bcast.

Ejercicios

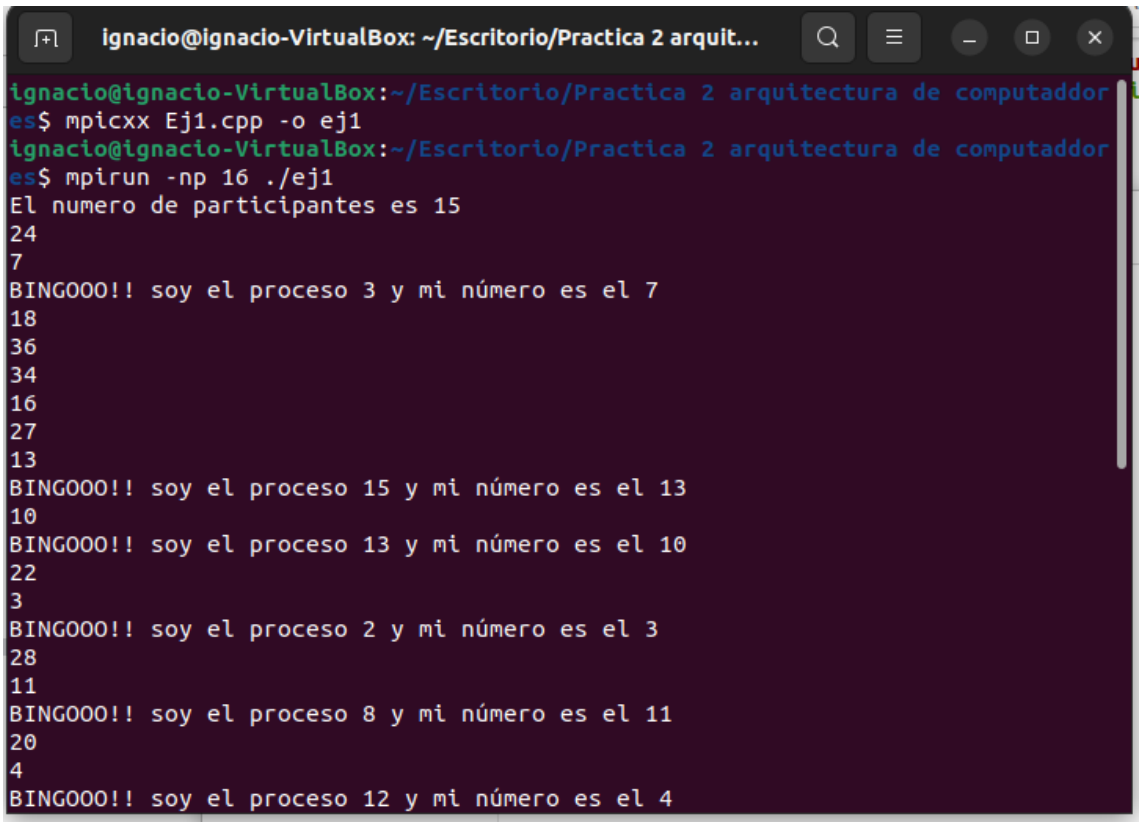
Ejercicio 1

Este ejercicio pide realizar un bingo, en el que el proceso 0 genera números aleatorios y el resto de los procesos toman un numero de un txt y cuando el proceso 0 diga su número cantaran bingo.

```
15 using namespace std;
16 int main(int argc, char ** argv) {
17   int rango, participantes, minum;
18   int contador = 1;
19   int bingo;
20   bool alwaystrue = true;
21   string linetext;
22   MPI_Init( & argc, & argv); //inicialización entorno MPI
23   MPI_Comm_rank(MPI_COMM_WORLD, & rango); //rango del proceso
24   MPI_Comm_size(MPI_COMM_WORLD, & participantes);
25   MPI_Request request; //variable de control
26
27   ifstream archivo("Cartones.txt");
28
29   if (rango == 0) {
30     cout << "El numero de participantes es " << participantes - 1 << endl; //le resto uno porque el
   proceso 0 no cuenta como parrticipante
31     sleep(2);
32     while (alwaystrue) {
33       bingo = 1 + rand() % (41 - 1);
34       cout << bingo << endl;
35       MPI_Bcast( & bingo, 1, MPI_INT, 0, MPI_COMM_WORLD);
36       sleep(1.5);
37     }
38
39   } else {
40     while (getline(archivo, linetext)) {
41
42       if (rango == contador) {
43         minum = stoi(linetext);
44
45       }
46
47       contador++;
48     }
49     while (alwaystrue) {
50       MPI_Bcast( & bingo, 1, MPI_INT, 0, MPI_COMM_WORLD); // al estar dentro de bucles infinitos
   tengo que hacer un broadcast para enviar y otro para recibir
51       if (minum == bingo) {
52         cout << "BINGOOO!! soy el proceso " << rango << " y mi número es el " << minum << endl;
53       }
54
55     }
56   }
57
58   MPI_Finalize(); //fin entorno MPI
59   return 0;
60 }
```

C++ Anchura del tabulador: 8 Ln 41, Col 1 INS

Solución

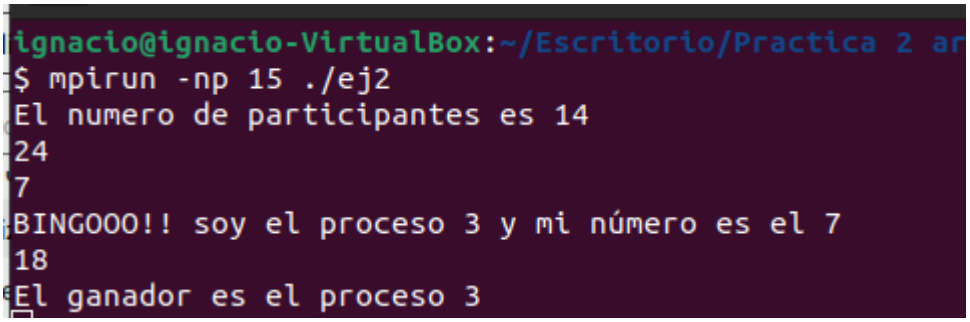

ignacio@ignacio-VirtualBox: ~/Escritorio/Practica 2 arquit...
ignacio@ignacio-VirtualBox:~/Escritorio/Practica 2 arquitectura de computaddor
es$ mpicxx Ej1.cpp -o ej1
ignacio@ignacio-VirtualBox:~/Escritorio/Practica 2 arquitectura de computaddor
es$ mpirun -np 16 ./ej1
El numero de participantes es 15
24
7
BINGOOO!! soy el proceso 3 y mi número es el 7
18
36
34
16
27
13
BINGOOO!! soy el proceso 15 y mi número es el 13
10
BINGOOO!! soy el proceso 13 y mi número es el 10
22
3
BINGOOO!! soy el proceso 2 y mi número es el 3
28
11
BINGOOO!! soy el proceso 8 y mi número es el 11
20
4
BINGOOO!! soy el proceso 12 y mi número es el 4

Ejercicio 2

Este segundo ejercicio pide básicamente finalizar de mejor manera el bingo, ya que el anterior era un bucle infinito, en este nos piden que cuando se cante bingo, se envíe una señal al proceso 0 y este anuncie el ganador y a su vez envíe otra señal al resto de los procesos para que se finalicen.

```
32
33   if (rango == 0) {
34     cout << "El numero de participantes es " << participantes - 1 << endl; //le resto uno porque el
   proceso 0 no cuenta como parrticipante
35     sleep(2);
36     while (alwaystrue) {
37     MPI_Status status;
38     int flag=0;
39         MPI_Iprobe(MPI_ANY_SOURCE, 0, MPI_COMM_WORLD, &flag,&status);
40     if(flag){
41     MPI_Recv(&rango, 1, MPI_INT, MPI_ANY_SOURCE, 0,MPI_COMM_WORLD, MPI_STATUS_IGNORE);
42     cout << "El ganador es el proceso "<< rango<< endl;
43     fin=0;
44
45     break;
46     }else{
47       bingo = 1 + rand() % (41 - 1);
48       cout << bingo << endl;
49       MPI_Bcast(&bingo, 1, MPI_INT, 0, MPI_COMM_WORLD);
50       sleep(2.5);
51     }
52     }
53     if(fin==0){
54         MPI_Bcast(&fin, 1, MPI_INT, 0, MPI_COMM_WORLD);
55     }
56     MPI_Finalize();
57     return 0;
58   } else {
59     while (getline(archivo, linetext)) {
60
61       if (rango == contador) {
62         minum = stoi(linetext);
63
64       }
65
66       contador++;
67     }
68     while (alwaystrue) {
69       MPI_Bcast(&bingo, 1, MPI_INT, 0, MPI_COMM_WORLD);
70
71       if (minum == bingo) {
72         cout << "BINGOOO!! soy el proceso " << rango << " y mi número es el " << minum << endl;
73         MPI_Send(&rango, 1,MPI_INT, 0, 0, MPI_COMM_WORLD);
74         break;
75       }
76     MPI_Status status;
77     int flag=0;
78     MPI_Iprobe(MPI_ANY_SOURCE, 0, MPI_COMM_WORLD, &flag,&status);
79       if(flag){
80 break;
81       }
82     }
```

Solución



En este caso no he podido completar el ejercicio, algunos de los problemas que me he encontrado han sido que al hacer el segundo brodcast para enviarle a todos los procesos un 0 como señal de que se tienen que cerrar no me lo leía. Al poner dos brodcast para recibir datos en el resto de los procesos simplemente no me leía ninguno, y al no poder indicar que se tenían que cerrar los procesos el programa nunca termina, no muestra nada infinito por pantalla, pero se queda en stand-by.

Conclusiones

Verdaderamente he aprendido bastante sobre como funciona la comunicación entre procesos, pero siento que me ha faltado un poco de información sobre el funcionamiento interno de MPI_Bcast para poder terminar perfectamente la práctica.