

3CV9 - Analysis and Design of Parallel Algorithms.

Práctica 3

Moreno González Gabriela.

October 2, 2017

Matrices de procesos

Nivel avanzado.

El código del programa es el siguiente:

```
#include <stdio.h>
#include <iostream>
#include <math.h>
#include <stdlib.h>
#include <time.h>
#include "mpi.h"

using namespace std;

#define TAMA 12
#define NP 9

int main(int argc, char** argv) {
    int rank, size, rank_cart,
        arriba, abajo, izq, der; //Vecinos en la topologia 2D.

    MPI_Init(&argc, &argv);
    MPI_Comm_size(MPI_COMM_WORLD, &size);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);

    MPI_Status status;
```

```

MPI_Comm COMM_CART;//Guardaremos el comunicador cartesiano.
MPI_Datatype subImagen;//Tipo de dato para una submatriz.

int dims[] = //tamaño de cada dimension del comunicador cartesiano
    {((int) sqrt(size)),
    size / ((int) sqrt(size))};
int periods[] = {0, 0};//No sera periodica ninguna dimension.
int tama_local = TAMA / dims[0];
int coord[2];//Podemos almacenar coordenadas 2D
int origen;//almacenaremos el rango del proceso 0 dentro del COMM_CART
int ImagenLocal[TAMA][TAMA];

if (size != NP && rank == 0) {
    cout << "El numero de procesos debe ser :" << NP << endl;
    MPI_Abort(MPI_COMM_WORLD, 1);
}

MPI_Cart_create(MPI_COMM_WORLD //A partir de los procesos de
COMM_WORLD.
    ,2 //Creamos una malla de 2 dimensiones.
    ,dims//Indicamos el tamaño de cada dimension.
    ,periods//Indicamos la periodicidad de cada dimension.
    ,true//Permitimos que MPI organice los procesos.
    //ya no tienen porque tener el mismo rango que antes.
    , &COMM_CART);//Guardamos el nuevo comunicador.

//Si tubieramos mas procesos que espacio en la malla recibira MPI_COMM_NULL
if (COMM_CART != MPI_COMM_NULL) {
    MPI_Comm_rank(COMM_CART, &rank_cart);//Obtenemos el nuevo rango

    //Creamos el tipo de dato subImagen

```

```

MPI_Type_vector(tama_local//numero de bloques
    ,tama_local//numero de enteros dentro de cada bloque
    ,TAMA//Cada una fila se coge un nuevo bloque
    ,MPI_INT//Cogen enteros
    ,&subImagen); //Guardamos el tipo de dato
MPI_Type_commit(&subImagen); //Confirmamos el tipo para poder usarlo.

```

```

if (rank == 0) { //Proceso 0 genera La imagen y la reparte
srand ( time(NULL) );
    for (int i = 0; i < TAMA; i++) {
        for (int j = 0; j < TAMA; j++) {
            ImagenLocal[i][j] = rand()%256; //Imagen en blanco y negro.
            cout << "[" << ImagenLocal[i][j] << " ";
        }
        cout << endl;
    }

    int aux;
    for (int i = 0; i < dims[0]; i++) //para cada proceso
        for (int j = 0; j < dims[1]; j++) { // se reparte su parte
            coord[0] = i;
            coord[1] = j;
            MPI_Cart_rank(COMM_CART//Para un proceso en este comunicador
                ,coord//con estas coordenadas
                , &aux); //Tiene el este rango.

            if (aux != rank_cart) {
                //enviamos las subimagenes.
                MPI_Send(&ImagenLocal[i * tama_local][j * tama_local],
                    1,
                    subImagen,

```

```

        aux,
        0,
        COMM_CART);
    }
}

MPI_Cart_coords(COMM_CART//Para un proceso en este comunicador
                ,rank_cart//con este rango
                , 2//como maximo en 2 coordenadas
                ,coord);//Dame sus coordenadas
} else {
    MPI_Cart_coords(COMM_CART, rank_cart, 2, coord);
    MPI_Recv(&ImagenLocal[tama_local * coord[0]][tama_local * coord[1]], 1,
subImagen, MPI_ANY_SOURCE, 0, COMM_CART, &status);
    origen = status.MPI_SOURCE;
}

}

//creamos este tipo de dato para almacenar columnas
MPI_Datatype vectorVertical;
MPI_Type_vector(tama_local, 1, TAMA, MPI_INT, &vectorVertical);
MPI_Type_commit(&vectorVertical);

MPI_Cart_shift(COMM_CART//en el comunicador cartesiano
               ,0 // en la dimension 0
               ,1 // con un desplazamiento de 1 sobre el actual
               ,&arriba //dame los vecinos (origen)
               ,&abajo);//dame los vecinos (destino)
//lo mismo para la dimension 1
MPI_Cart_shift(COMM_CART, 1, 1, &izq, &der);

```

```

if (arriba != MPI_PROC_NULL)//si tengo vecino arriba mando mi fila superior
    MPI_Send(&ImagenLocal[tama_local * coord[0]]
        [tama_local * coord[1]], tama_local, MPI_INT, arriba, 0, COMM_CART);

if (abajo != MPI_PROC_NULL)//si tengo vecino abajo recibo su fila superior
    MPI_Recv(&ImagenLocal[(tama_local * coord[0]) + tama_local]
        [tama_local * coord[1]], tama_local, MPI_INT, abajo, 0, COMM_CART, &status);

if (abajo != MPI_PROC_NULL)//si tengo vecino abajo mando mi fila inferior
    MPI_Send(&ImagenLocal[(tama_local * coord[0]) + tama_local - 1]
        [tama_local * coord[1]], tama_local, MPI_INT, abajo, 0, COMM_CART);

if (arriba != MPI_PROC_NULL)//si tengo vecino arriba recibo su fila inferior
    MPI_Recv(&ImagenLocal[(tama_local * coord[0]) - 1]
        [tama_local * coord[1]], tama_local, MPI_INT, arriba, 0, COMM_CART, &status);

if (izq != MPI_PROC_NULL)//si tengo vecino izquierda mando mi columna izquierda.
    MPI_Send(&ImagenLocal[tama_local * coord[0]]
        [tama_local * coord[1]], 1, vectorVertical, izq, 0, COMM_CART);

if (der != MPI_PROC_NULL)//si tengo vecino derecha recibo su columna derecha.
    MPI_Recv(&ImagenLocal[tama_local * coord[0]]
        [(tama_local * coord[1]) + tama_local], 1, vectorVertical, der, 0, COMM_CART,
&status);

if (der != MPI_PROC_NULL)//si tengo vecino derecha mando mi columna derecha
    MPI_Send(&ImagenLocal[tama_local * coord[0]]
        [(tama_local * coord[1]) + tama_local - 1], 1, vectorVertical, der, 0,
COMM_CART);

if (izq != MPI_PROC_NULL)//si tengo vecino izquierda recibo su columna derecha.

```

```

MPI_Recv(&ImagenLocal[tama_local * coord[0]]
        [(tama_local * coord[1]) - 1], 1, vectorVertical, izq, 0, COMM_CART, &status);

//APLICAMOS el filtro localmente.
int local[tama_local][tama_local];
for (int i = 0; i < tama_local; i++) {
    for (int j = 0; j < tama_local; j++) {
        int nuevo = ImagenLocal[tama_local * coord[0] + i][tama_local * coord[1] +
j]*2;
        int div = 2;
        if (coord[0] != 0) { //Arriba
            nuevo += ImagenLocal[tama_local * coord[0] + i - 1][tama_local * coord[1] +
j];
            div++;
        }
        if (coord[0] < dims[0] - 1) { //Abajo
            nuevo += ImagenLocal[tama_local * coord[0] + i + 1][tama_local * coord[1] +
j];
            div++;
        }
        if (coord[1] != 0) { //Izquierda
            nuevo += ImagenLocal[tama_local * coord[0] + i][tama_local * coord[1] + j
- 1];
            div++;
        }
        if (coord[1] < dims[0] - 1) { //Derecha
            nuevo += ImagenLocal[tama_local * coord[0] + i][tama_local * coord[1] + j +
1];
            div++;
        }
        local[i][j] = (nuevo / div) % 256;
    }
}

```

```

    }
}

if (rank != 0) { //Si no soy el proceso 0 mando mi matriz
    MPI_Send(local, tama_local*tama_local, MPI_INT, origen, 0, COMM_CART);
} else { //Si soy el 0 recibo las partes de los demas procesos.
    for (int i = 0; i < tama_local; i++)
        for (int j = 0; j < tama_local; j++)
            ImagenLocal[(tama_local * coord[0]) + i][(tama_local * coord[1]) + j] =
local[i][j];
    for (int i = 0; i < size - 1; i++) {
        MPI_Probe(MPI_ANY_SOURCE, MPI_ANY_TAG, COMM_CART, &status);
        MPI_Cart_coords(COMM_CART, status.MPI_SOURCE, 2, coord);
        MPI_Recv(&ImagenLocal[tama_local * coord[0]]
[tama_local * coord[1]], 1, subImagen, status.MPI_SOURCE, 0,
COMM_CART,
&status);

    }
    cout << "++++++RESULTADO++++++" << endl;
    for (int i = 0; i < TAMA; i++) {
        for (int j = 0; j < TAMA; j++) {
            cout << "[" << ImagenLocal[i][j] << "];"
        }
        cout << endl;
    }
}

MPI_Type_free(&subImagen); //liberamos el tipo de dato subImagen
MPI_Type_free(&vectorVertical); //liberamos el tipo dato vectorVertical

```

```

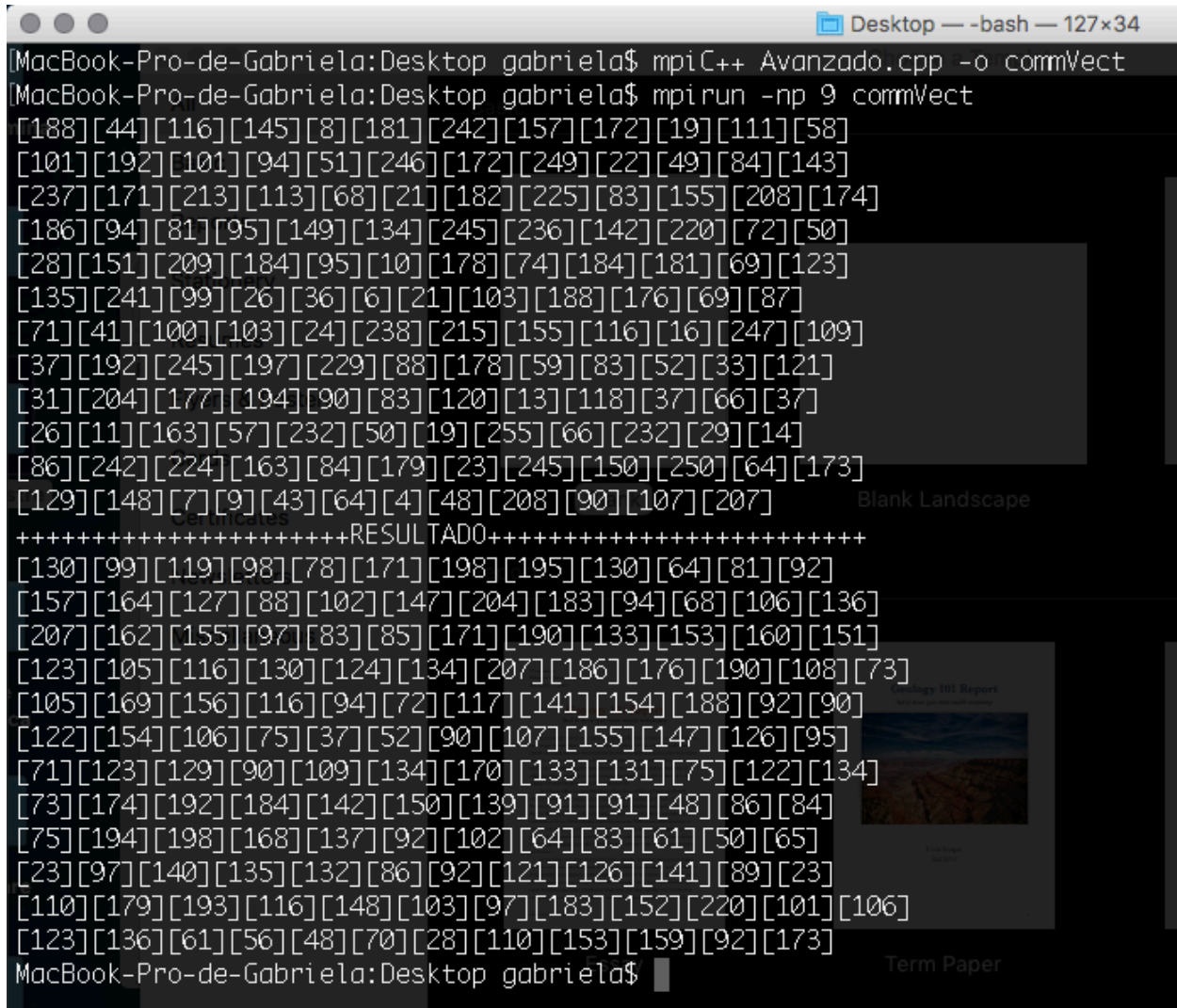
MPI_Finalize();

return 0;

}

```

Ahora, para compilar y ejecutarlo se emplea de la siguiente manera en MAC:



```

MacBook-Pro-de-Gabriela:Desktop gabriela$ mpiC++ Avanzado.cpp -o commVect
MacBook-Pro-de-Gabriela:Desktop gabriela$ mpirun -np 9 commVect
[188][44][116][145][8][181][242][157][172][19][111][58]
[101][192][101][94][51][246][172][249][22][49][84][143]
[237][171][213][113][68][21][182][225][83][155][208][174]
[186][94][81][95][149][134][245][236][142][220][72][50]
[28][151][209][184][95][10][178][74][184][181][69][123]
[135][241][99][26][36][6][21][103][188][176][69][87]
[71][41][100][103][24][238][215][155][116][16][247][109]
[37][192][245][197][229][88][178][59][83][52][33][121]
[31][204][177][194][90][83][120][13][118][37][66][37]
[26][11][163][57][232][50][19][255][66][232][29][14]
[86][242][224][163][84][179][23][245][150][250][64][173]
[129][148][7][9][43][64][4][48][208][90][107][207]
*****RESULTADO*****
[130][99][119][98][78][171][198][195][130][64][81][92]
[157][164][127][88][102][147][204][183][94][68][106][136]
[207][162][155][97][83][85][171][190][133][153][160][151]
[123][105][116][130][124][134][207][186][176][190][108][73]
[105][169][156][116][94][72][117][141][154][188][92][90]
[122][154][106][75][37][52][90][107][155][147][126][95]
[71][123][129][90][109][134][170][133][131][75][122][134]
[73][174][192][184][142][150][139][91][91][48][86][84]
[75][194][198][168][137][92][102][64][83][61][50][65]
[23][97][140][135][132][86][92][121][126][141][89][23]
[110][179][193][116][148][103][97][183][152][220][101][106]
[123][136][61][56][48][70][28][110][153][159][92][173]
MacBook-Pro-de-Gabriela:Desktop gabriela$

```

Haciendo una segunda prueba de escritorio:


```

MacBook-Pro-de-Gabriela:Desktop gabriela$ mpirun -np 9 commVect
[67][233][77][85][134][157][8][215][203][203][254][217]
[180][16][190][61][155][49][59][36][73][232][38][250]
[156][244][24][98][238][149][26][155][243][109][56][190]
[76][236][233][196][24][192][155][146][115][188][17][102]
[196][169][24][205][155][43][30][7][82][132][137][18]
[148][146][142][5][61][243][131][142][221][198][160][212]
[174][198][117][127][194][24][128][51][229][115][153][58]
[14][77][181][225][192][32][208][241][145][248][73][61]
[200][29][217][123][41][59][208][171][1][241][153][237]
[206][234][147][167][97][51][106][202][242][238][135][101]
[145][80][158][139][130][167][208][34][34][141][131][90]
[105][222][49][131][58][137][78][228][101][64][119][18]
+++++
*****RESULTADO*****
[136][139][107][91][133][101][89][135][173][210][187][234]
[133][116][116][93][131][92][45][71][106][161][91][182]
[158][187][94][157][149][150][102][145][189][162][59][134]
[146][218][171][155][118][121][135][113][114][155][89][59]
[157][148][125][153][107][117][66][69][101][146][116][97]
[162][160][86][80][119][124][134][115][179][172][161][132]
[141][147][136][135][132][107][111][140][175][181][130][108]
[95][112][184][178][146][91][170][176][152][199][140][98]
[110][88][184][128][91][79][170][158][79][182][155][172]
[211][161][169][138][90][72][134][184][171][239][165][143]
[144][138][150][143][132][144][144][102][86][138][134][103]
[144][143][96][114][102][115][145][133][116][92][108][61]
MacBook-Pro-de-Gabriela:Desktop gabriela$

```