Práctica 2: Implementación distribuida de un algoritmo paralelo de datos:

UNIVERSIDAD DE GRANADA

Luis González Romero, XXXXXXXXX, luisgonromero@correo.ugr.es

Escuela Técnica Superior de Ingeniería informática y Telecomunicaciones

29 de junio de 2021

1. Descomposición bidimensional (por bloques 2D)

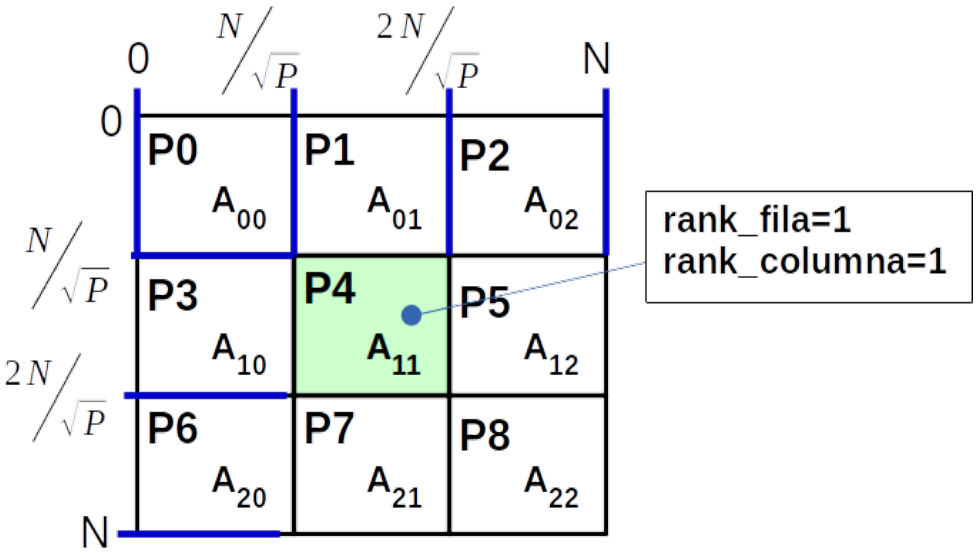

Figura 1: Distribución por bloques 2D de la matriz *A*

Este reparto de la matriz se hace empaquetando la matriz de la forma que ilustra la siguiente figura:

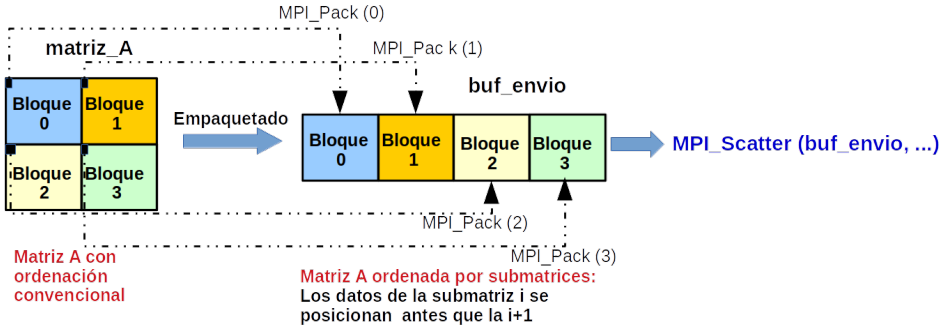

Figura 2: Empaquetado de la matriz por bloques para p=4

```
MPI_Type_vector(tam, tam, n, MPI_LONG, &MPI_BLOQUE);
MPI_Type_commit(&MPI_BLOQUE);
int comienzo, posicion=0;
for(int i=0; i<p; ++i)
{
  row_p = i/raiz_p;
  col_p = i%raiz_p;
  comienzo = col_p*tam + row_p*tam*tam*raiz_p;
  MPI_Pack(&A[row_p*tam][col_p*tam], 1, MPI_BLOQUE, buf_envio, sizeof(long)*n*n, &posicion, MPI_COMM_WORLD);
}
MPI_Type_free(&MPI_BLOQUE);
```

Figura 3: Código para realizar el empaquetado de *A*

Los comunicadores necesarios para la práctica se han obtenido de la siguiente manera.

```
int raiz_p = sqrt(p), tam = n/raiz_p;
int row_color  = rank/raiz_p,
    col_color  = rank%raiz_p;
MPI_Comm diag_comm;
int diag_color=(row_color == col_color) ? 0 : 1, diag_p, diag_rank;
MPI_Comm row_comm, col_comm;
MPI_Comm_split(MPI_COMM_WORLD, row_color, rank, &row_comm);
MPI_Comm_split(MPI_COMM_WORLD, col_color, rank, &col_comm);
MPI_Comm_split(MPI_COMM_WORLD, diag_color, rank, &diag_comm);

int row_rank, col_rank, row_p, col_p;
MPI_Comm_rank(row_comm, &row_rank);
MPI_Comm_size(row_comm, &row_p);
MPI_Comm_rank(col_comm, &col_rank);
MPI_Comm_size(col_comm, &col_p);
MPI_Comm_rank(diag_comm, &diag_rank);
MPI_Comm_size(diag_comm, &diag_p);
```

Figura 4: Obtención de los comunicadores necesarios

- *Comunicador para las filas*: las partes enteras nos divide el comunicador en filas
- *Comunicador para las columnas*: el módulo nos divide a los procesadores en columnas con respecto a $\sqrt{P}$
- *Comunicador para la diagonal*: se divide dependiendo de si el comunicador para la fila y la columna coinciden.

Para repartir la matriz A, se realiza el siguiente *scatter*.

```
long *buf_recep = new long[tam*tam];
MPI_Scatter(buf_envio, sizeof(long)*tam*tam, MPI_PACKED, buf_recep, tam*tam, MPI_LONG, 0, MPI_COMM_WORLD);
```

Figura 5: Código del reparto de la matriz A en bloques2D

A continuación, se realiza el *scatter* y el *broadcast* de x para obtener los elementos parciales, tal y como se indica en la siguiente figura:

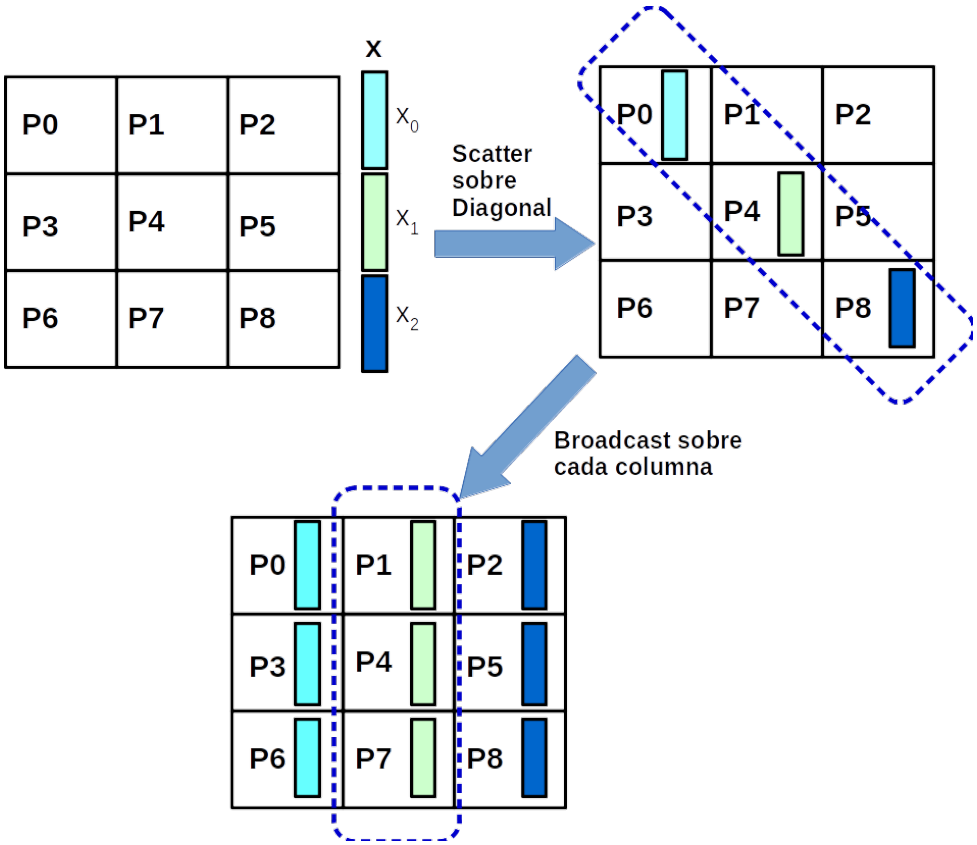

Figura 6: Distribución de los subvectores del vector x en una malla de ejemplo

```
long *xlocal = new long [tam];
MPI_Scatter(x, tam, MPI_LONG, xlocal, tam, MPI_LONG, 0, diag_comm);
MPI_Bcast(xlocal, tam, MPI_LONG, row_rank, col_comm);
```

Figura 7: Código que simula el comportamiento de la imagen anterior

Pasamos a calcular la parte que le corresponde del vector de salida y para cada proceso.

```
long *ylocal = new long[tam];
for(int i=0; i<tam; ++i)
{
  ylocal[i] = 0;
  for(int j=0; j<tam; ++j)
  {
    ylocal[i] += buf_recep[j+tam*i] * xlocal[j];
  }
}
```

Figura 8: Código que calcula el segmento de y correspondiente

Y finalmente, obtenemos las sumas locales(mediante reducción) y las reunimos en el proceso 0.

```
long *y_row_reduce = new long[tam];

MPI_Reduce(ylocal, y_row_reduce, tam, MPI_LONG, MPI_SUM, col_rank, row_comm);
long *y_gather = new long[n];

MPI_Gather(y_row_reduce, tam, MPI_LONG, y_gather, tam, MPI_LONG, 0, diag_comm);
MPI_Finalize();
```

Figura 9: Código de la obtención de y

Ejecución de ejemplo:

```
roronoasins in ~/Documents/ppr/practicas/p2 λ mpirun -np 4 mult-mv2d 4
Invalid MIT-MAGIC-COOKIE-1 keyLa matriz y el vector generados son
[2  977  26  671]         [29]
[740  634  448  813]      [52]
[852  731  300  667]      [82]
[749  923  939  314]      [21]

El resultado obtenido y el esperado son:
        67085   |       67085
        108237  |       108237
        101327  |       101327
        153309  |       153309
No hubo errores
El tiempo tardado ha sido 7.526e-06 segundos.
```

Figura 10: Salida de ejemplo para una matriz 4x4 con p=4

Medición de los distintos ejercicios:

		Unidimensional(bloques de filas)		Bidimensional(bloques 2d)	
N	secuencial	P=4	Ganancia	P=4	Ganancia
300	0,000935	0,000214529	4,35838511343455	0,0001501406	6,22749609366154
600	0,002398	0,000746907	3,21057373943476	0,0005137082	4,66801970457158
900	0,0039364	0,001480536	2,65876682498771	0,001214088	3,2422690941678
1200	0,0053216	0,001930654	2,75637167509041	0,001607496	3,31049035269761
1400	0,008318	0,002544616	3,26886257101268	0,001906264	4,3635089368524

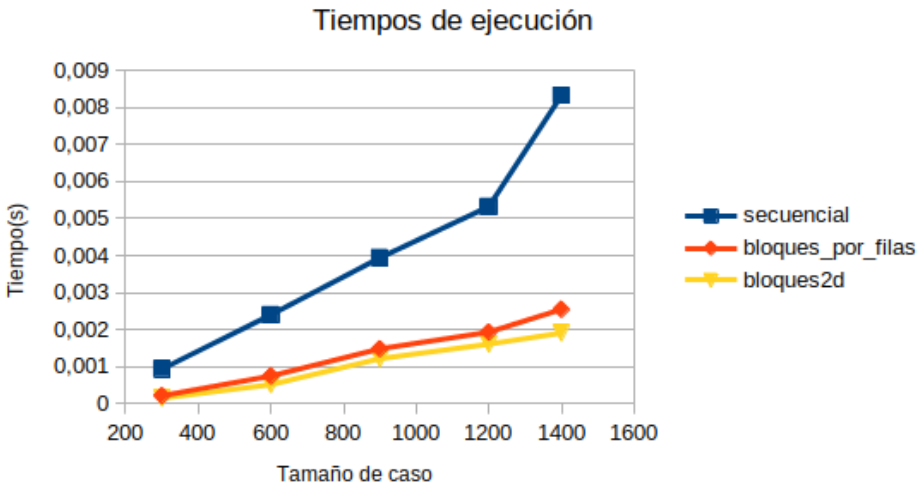

Figura 11: Tiempos de ejecución obtenidos y ganancia