

MÉTODOS NUMÉRICOS PARA LA COMPUTACIÓN

Tema 7: Programación MPI

2020/21

21 de diciembre de 2020

Grupo 03: José María Amusquívar Poppe y Prashant Jeswani Tejawani

Universidad de Las Palmas de Gran Canaria

Escuela de Ingeniería en Informática

Índice

Actividad práctica 1	3
Actividad práctica 2	4
Actividad práctica 3	6
Actividad práctica 4	7
Actividad práctica 5	8
Actividad práctica 6	9
Actividad práctica 7	10
Actividad práctica 8	11
Actividad práctica 9	13
Actividad práctica 10	14

Actividad práctica 1

Se ha ejecutado el siguiente código el cual genera procesos que muestran un mensaje por pantalla:

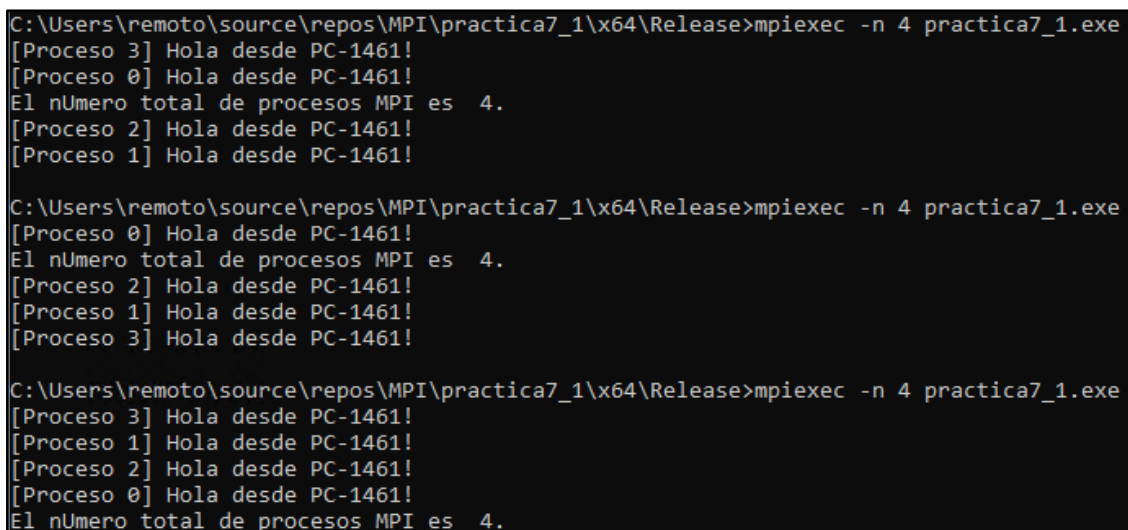
```
#include "mpi.h"
#include <stdio.h>

void main(int argc, char *argv[])
{
    int rank, size, length;
    char name[MPI_MAX_PROCESSOR_NAME];

    MPI_Init(&argc, &argv);
    MPI_Get_processor_name(name, &length);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    MPI_Comm_size(MPI_COMM_WORLD, &size);
    printf("[Proceso %d] Hola desde %s!\n", rank, name);
    if (rank == 0) printf("El número total de procesos MPI es %d.\n", size);
    MPI_Finalize();
    return;
}
```

Se ejecuta el programa cambiando el número de procesos; para ello hay que usar la línea de comandos: `mpiexec.exe -n 2 miProyecto.exe`

A continuación, se ejecuta el programa varias veces con un mismo número de procesos con el fin de comprobar que el resultado no es determinista, ya que el orden en el que aparecen los mensajes varía en cada ejecución.



```
C:\Users\remoto\source\repos\MPI\practica7_1\x64\Release>mpiexec -n 4 practica7_1.exe
[Proceso 3] Hola desde PC-1461!
[Proceso 0] Hola desde PC-1461!
El número total de procesos MPI es 4.
[Proceso 2] Hola desde PC-1461!
[Proceso 1] Hola desde PC-1461!

C:\Users\remoto\source\repos\MPI\practica7_1\x64\Release>mpiexec -n 4 practica7_1.exe
[Proceso 0] Hola desde PC-1461!
El número total de procesos MPI es 4.
[Proceso 2] Hola desde PC-1461!
[Proceso 1] Hola desde PC-1461!
[Proceso 3] Hola desde PC-1461!

C:\Users\remoto\source\repos\MPI\practica7_1\x64\Release>mpiexec -n 4 practica7_1.exe
[Proceso 3] Hola desde PC-1461!
[Proceso 1] Hola desde PC-1461!
[Proceso 2] Hola desde PC-1461!
[Proceso 0] Hola desde PC-1461!
El número total de procesos MPI es 4.
```

Figura 1. Ejecución empleando 4 hilos.

Comprobamos que efectivamente el resultado obtenido no es determinista y el orden de los hilos varía en cada ejecución.

Actividad práctica 2

Posteriormente, para medir el tiempo que tarda una operación:

```
double tinicial = MPI_Wtime();  
  
...  
double tfinal = MPI_Wtime();  
double tiempo = tfinal - tinicial;
```

Por lo que se modifica el programa anterior para que cada proceso realice una tarea que consuma tiempo como, por ejemplo, multiplicar dos números en coma flotante varios millones de veces. Además, se añade al mensaje que muestra cada proceso el tiempo que ha tardado en ejecutar la operación.

Se han realizado varias pruebas cambiando el número total de procesos y para determinar si, a partir de los datos obtenidos, podemos verificar el número de procesos que es capaz de ejecutar el procesador de forma simultánea.

```
C:\Users\remoto\source\repos\MPI\practica7_1\x64\Release>mpiexec -n 2 practica7_1.exe  
[Proceso 1] 0.01918400 s!, el resultado es inf  
[Proceso 0] 0.01918330 s!, el resultado es inf  
  
C:\Users\remoto\source\repos\MPI\practica7_1\x64\Release>mpiexec -n 4 practica7_1.exe  
[Proceso 1] 0.01925230 s!, el resultado es inf  
[Proceso 2] 0.01923980 s!, el resultado es inf  
[Proceso 0] 0.01925550 s!, el resultado es inf  
[Proceso 3] 0.01926260 s!, el resultado es inf  
  
C:\Users\remoto\source\repos\MPI\practica7_1\x64\Release>mpiexec -n 6 practica7_1.exe  
[Proceso 5] 0.01930500 s!, el resultado es inf  
[Proceso 2] 0.01931540 s!, el resultado es inf  
[Proceso 3] 0.01937210 s!, el resultado es inf  
[Proceso 4] 0.01929900 s!, el resultado es inf  
[Proceso 1] 0.01942360 s!, el resultado es inf  
[Proceso 0] 0.01940760 s!, el resultado es inf  
  
C:\Users\remoto\source\repos\MPI\practica7_1\x64\Release>mpiexec -n 8 practica7_1.exe  
[Proceso 2] 0.01923990 s!, el resultado es inf  
[Proceso 1] 0.01912310 s!, el resultado es inf  
[Proceso 6] 0.01968170 s!, el resultado es inf  
[Proceso 0] 0.01939570 s!, el resultado es inf  
[Proceso 4] 0.01965390 s!, el resultado es inf  
[Proceso 5] 0.01920200 s!, el resultado es inf  
[Proceso 7] 0.02185780 s!, el resultado es inf  
[Proceso 3] 0.02001150 s!, el resultado es inf
```

Figura 2. Ejecución empleando hasta 8 hilos simultáneos.

```

C:\Users\remoto\source\repos\MPI\practica7_1\x64\Release>mpiexec -n 9 practica7_1.exe
[Proceso 4] 0.01905890 s!, el resultado es inf
[Proceso 7] 0.03676550 s!, el resultado es inf
[Proceso 3] 0.01907610 s!, el resultado es inf
[Proceso 2] 0.01916110 s!, el resultado es inf
[Proceso 0] 0.01903010 s!, el resultado es inf
[Proceso 1] 0.01915270 s!, el resultado es inf
[Proceso 5] 0.01900850 s!, el resultado es inf
[Proceso 6] 0.01900360 s!, el resultado es inf
[Proceso 8] 0.01900350 s!, el resultado es inf

C:\Users\remoto\source\repos\MPI\practica7_1\x64\Release>mpiexec -n 10 practica7_1.exe
[Proceso 0] 0.01900320 s!, el resultado es inf
[Proceso 5] 0.01903350 s!, el resultado es inf
[Proceso 6] 0.02897660 s!, el resultado es inf
[Proceso 9] 0.01900340 s!, el resultado es inf
[Proceso 7] 0.02895740 s!, el resultado es inf
[Proceso 3] 0.01900250 s!, el resultado es inf
[Proceso 8] 0.01900610 s!, el resultado es inf
[Proceso 1] 0.01900250 s!, el resultado es inf
[Proceso 2] 0.01901260 s!, el resultado es inf
[Proceso 4] 0.01897610 s!, el resultado es inf

```

Figura 3. Ejecución empleando 9 y 10 hilos.

Por tanto, se comprueba que, a partir de los 9 hilos simultáneos, existen hilos que sobrepasan la media temporal de ejecución, por lo que se puede determinar que el procesador, únicamente, es capaz de ejecutar 8 hilos de manera simultánea.

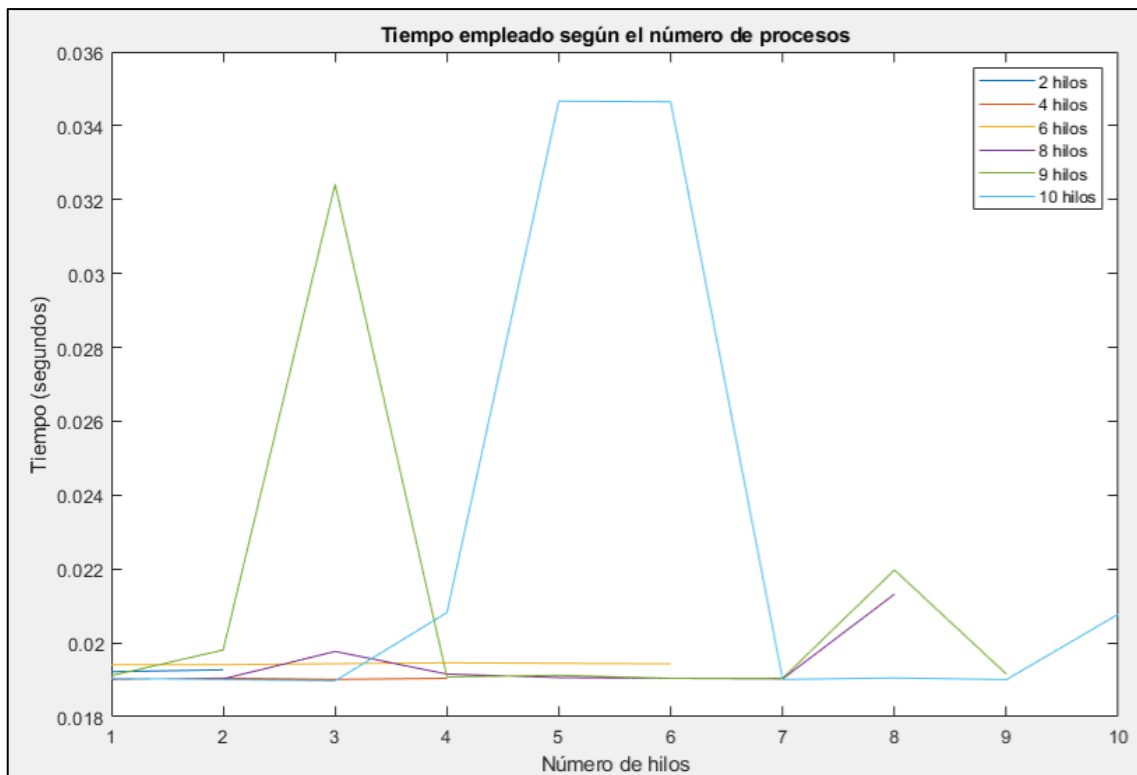


Figura 4. Gráfica que demuestra el máximo número de hilos simultáneos.

Actividad práctica 3

Se modifica el código anterior para que, asumiendo que el número de procesos es par, cada proceso elija otro proceso como compañero:

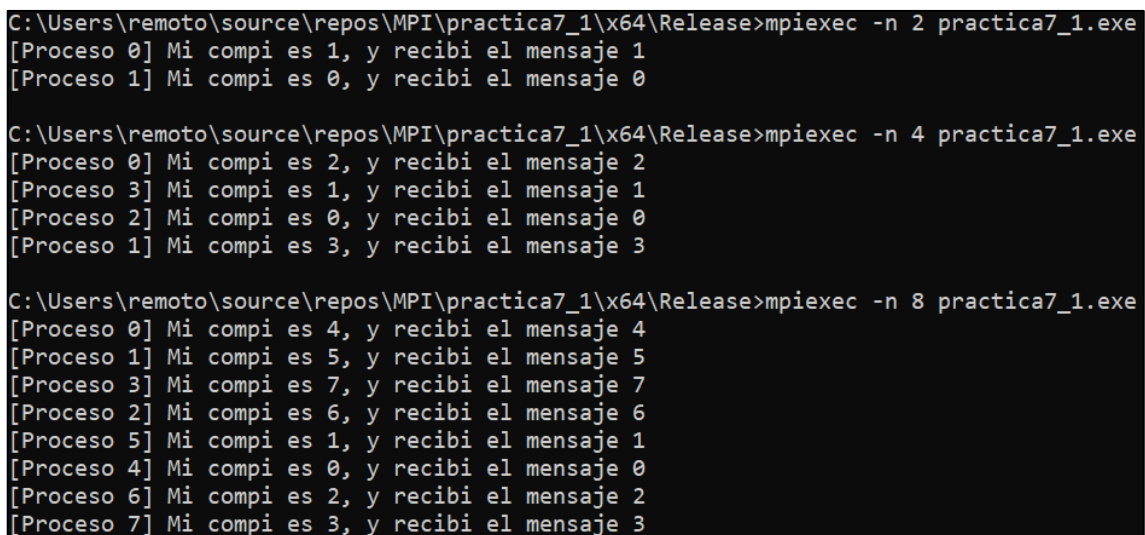
```
si (rango < número_procesos/2)      compañero = rango + número_procesos/2
si no                                compañero = rango - número_procesos/2
```

Tras elegir un compañero, cada proceso debe enviarle un mensaje con su identificador y recibir el mensaje recíproco que enviará su compañero:

```
MPI_Status status;
int rank, partner, message;

MPI_Send(&rank, 1, MPI_INT, partner, 1, MPI_COMM_WORLD);
MPI_Recv(&message, 1, MPI_INT, partner, 1, MPI_COMM_WORLD, &status);
```

Los procesos muestran un segundo mensaje por pantalla que indique quién es su compañero y qué mensaje ha recibido de él.



```
C:\Users\remoto\source\repos\MPI\practica7_1\x64\Release>mpiexec -n 2 practica7_1.exe
[Proceso 0] Mi compi es 1, y recibí el mensaje 1
[Proceso 1] Mi compi es 0, y recibí el mensaje 0

C:\Users\remoto\source\repos\MPI\practica7_1\x64\Release>mpiexec -n 4 practica7_1.exe
[Proceso 0] Mi compi es 2, y recibí el mensaje 2
[Proceso 3] Mi compi es 1, y recibí el mensaje 1
[Proceso 2] Mi compi es 0, y recibí el mensaje 0
[Proceso 1] Mi compi es 3, y recibí el mensaje 3

C:\Users\remoto\source\repos\MPI\practica7_1\x64\Release>mpiexec -n 8 practica7_1.exe
[Proceso 0] Mi compi es 4, y recibí el mensaje 4
[Proceso 1] Mi compi es 5, y recibí el mensaje 5
[Proceso 3] Mi compi es 7, y recibí el mensaje 7
[Proceso 2] Mi compi es 6, y recibí el mensaje 6
[Proceso 5] Mi compi es 1, y recibí el mensaje 1
[Proceso 4] Mi compi es 0, y recibí el mensaje 0
[Proceso 6] Mi compi es 2, y recibí el mensaje 2
[Proceso 7] Mi compi es 3, y recibí el mensaje 3
```

Figura 5. Ejecución con varios hilos, comprobando su funcionamiento.

Observamos que los mensajes son enviados y recibidos correctamente para distintos número de hilos.

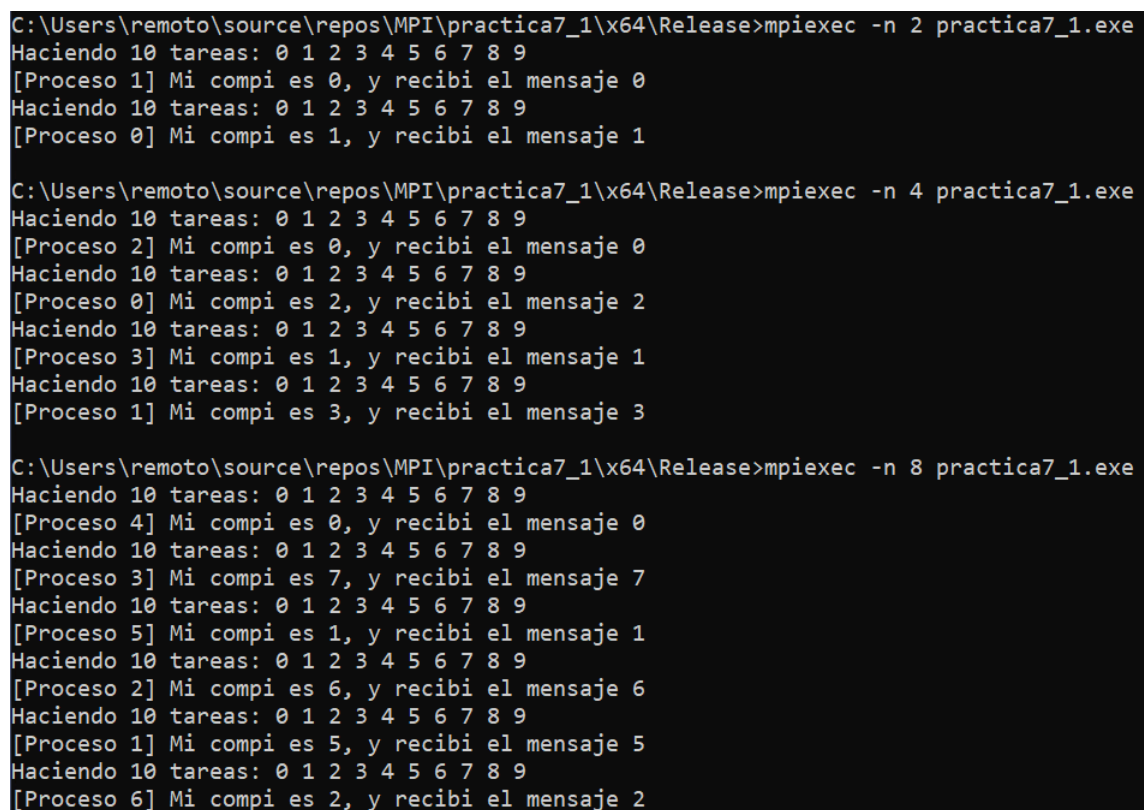
Actividad práctica 4

Se modifica el código anterior para que haga uso de comunicaciones no bloqueantes; para ello será necesario que la variable *status* sea un vector de dos elementos y habrá que añadir otro vector para las peticiones:

```
MPI_Status status[2];  
MPI_Request request[2];
```

Tras realizar el envío y la recepción se puede llevar a cabo cualquier tarea que no use los datos implicados en la operación de comunicación, pero para dar por finalizada la operación, hay que verificar que ha terminado:

```
MPI_Isend(&rank, 1, MPI_INT, partner, 1, MPI_COMM_WORLD, &request[1]);  
MPI_Irecv(&message, 1, MPI_INT, partner, 1, MPI_COMM_WORLD, &request[0]);  
...  
MPI_Waitall(2, request, status);
```



```
C:\Users\remoto\source\repos\MPI\practica7_1\x64\Release>mpiexec -n 2 practica7_1.exe  
Haciendo 10 tareas: 0 1 2 3 4 5 6 7 8 9  
[Proceso 1] Mi compi es 0, y recibí el mensaje 0  
Haciendo 10 tareas: 0 1 2 3 4 5 6 7 8 9  
[Proceso 0] Mi compi es 1, y recibí el mensaje 1  
  
C:\Users\remoto\source\repos\MPI\practica7_1\x64\Release>mpiexec -n 4 practica7_1.exe  
Haciendo 10 tareas: 0 1 2 3 4 5 6 7 8 9  
[Proceso 2] Mi compi es 0, y recibí el mensaje 0  
Haciendo 10 tareas: 0 1 2 3 4 5 6 7 8 9  
[Proceso 0] Mi compi es 2, y recibí el mensaje 2  
Haciendo 10 tareas: 0 1 2 3 4 5 6 7 8 9  
[Proceso 3] Mi compi es 1, y recibí el mensaje 1  
Haciendo 10 tareas: 0 1 2 3 4 5 6 7 8 9  
[Proceso 1] Mi compi es 3, y recibí el mensaje 3  
  
C:\Users\remoto\source\repos\MPI\practica7_1\x64\Release>mpiexec -n 8 practica7_1.exe  
Haciendo 10 tareas: 0 1 2 3 4 5 6 7 8 9  
[Proceso 4] Mi compi es 0, y recibí el mensaje 0  
Haciendo 10 tareas: 0 1 2 3 4 5 6 7 8 9  
[Proceso 3] Mi compi es 7, y recibí el mensaje 7  
Haciendo 10 tareas: 0 1 2 3 4 5 6 7 8 9  
[Proceso 5] Mi compi es 1, y recibí el mensaje 1  
Haciendo 10 tareas: 0 1 2 3 4 5 6 7 8 9  
[Proceso 2] Mi compi es 6, y recibí el mensaje 6  
Haciendo 10 tareas: 0 1 2 3 4 5 6 7 8 9  
[Proceso 1] Mi compi es 5, y recibí el mensaje 5  
Haciendo 10 tareas: 0 1 2 3 4 5 6 7 8 9  
[Proceso 6] Mi compi es 2, y recibí el mensaje 2
```

Figura 6. Ejecución empleando varios hilos, realizando una tarea en segundo plano.

Podemos observar que se han enviado y recibido correctamente los mensajes haciendo uso de comunicaciones no bloqueantes, llevando a cabo cualquier tarea que no use los datos implicados.

Actividad práctica 5

El código anterior envía como mensaje un número entero, esto se ha modificado para que envíe una cadena de caracteres con un pequeño texto.

El texto es distinto en cada proceso, para ello, se añade el identificador del proceso que lo envía con el fin de identificar y verificar que el mensaje es correcto.

```
C:\Users\remoto\source\repos\MPI\practica7_1\x64\Release>mpiexec -n 4 practica7_1.exe
Haciendo 10 tareas: 0 1 2 3 4 5 6 7 8 9
[Proceso 1] Mi compi es 3, y recibí el mensaje 3 - Hola pepito
Haciendo 10 tareas: 0 1 2 3 4 5 6 7 8 9
[Proceso 0] Mi compi es 2, y recibí el mensaje 2 - Hola pepito
Haciendo 10 tareas: 0 1 2 3 4 5 6 7 8 9
[Proceso 3] Mi compi es 1, y recibí el mensaje 1 - Hola pepito
Haciendo 10 tareas: 0 1 2 3 4 5 6 7 8 9
[Proceso 2] Mi compi es 0, y recibí el mensaje 0 - Hola pepito

C:\Users\remoto\source\repos\MPI\practica7_1\x64\Release>mpiexec -n 8 practica7_1.exe
Haciendo 10 tareas: 0 1 2 3 4 5 6 7 8 9
[Proceso 5] Mi compi es 1, y recibí el mensaje 1 - Hola pepito
Haciendo 10 tareas: 0 1 2 3 4 5 6 7 8 9
[Proceso 2] Mi compi es 6, y recibí el mensaje 6 - Hola pepito
Haciendo 10 tareas: 0 1 2 3 4 5 6 7 8 9
[Proceso 0] Mi compi es 4, y recibí el mensaje 4 - Hola pepito
Haciendo 10 tareas: 0 1 2 3 4 5 6 7 8 9
[Proceso 1] Mi compi es 5, y recibí el mensaje 5 - Hola pepito
Haciendo 10 tareas: 0 1 2 3 4 5 6 7 8 9
[Proceso 3] Mi compi es 7, y recibí el mensaje 7 - Hola pepito
Haciendo 10 tareas: 0 1 2 3 4 5 6 7 8 9
[Proceso 4] Mi compi es 0, y recibí el mensaje 0 - Hola pepito
Haciendo 10 tareas: 0 1 2 3 4 5 6 7 8 9
[Proceso 6] Mi compi es 2, y recibí el mensaje 2 - Hola pepito
Haciendo 10 tareas: 0 1 2 3 4 5 6 7 8 9
[Proceso 7] Mi compi es 3, y recibí el mensaje 3 - Hola pepito
```

Figura 7. Ejecución de varios hilos, enviando un mensaje de texto y realizando una operación en segundo plano.

Una vez más, observamos que los mensajes se han mandado y recibido de forma correcta con un mensaje junto al identificador del emisor.

Actividad práctica 6

A continuación, se implementa un programa en el que el proceso de rango cero, que actuará como proceso principal, inicializa una cadena de caracteres con un determinado texto.

Tras esto, se ejecuta una operación de difusión (*broadcast*) para que todos los procesos reciban el mensaje; los procesos imprimen el mensaje por pantalla para verificar que se ha recibido correctamente.

Se calcula el tiempo que tarda en realizarse la operación y se compara con el tiempo que tarda en realizarse si se realiza con comunicaciones punto a punto; se ha tenido que aumentar el tamaño del mensaje para que el tiempo empleado sea relevante.

```
C:\Users\remoto\source\repos\MPI\practica7_1\x64\Release>mpiexec -n 2 practica7_1.exe
[Proceso 1] He recibido del proceso 0, el mensaje de longitud 480000, con tiempo: 0.002532 s.
[Proceso 0] He recibido del proceso 0, el mensaje de longitud 480000, con tiempo: 0.002557 s.

C:\Users\remoto\source\repos\MPI\practica7_1\x64\Release>mpiexec -n 4 practica7_1.exe
[Proceso 1] He recibido del proceso 0, el mensaje de longitud 480000, con tiempo: 0.002517 s.
[Proceso 3] He recibido del proceso 0, el mensaje de longitud 480000, con tiempo: 0.002732 s.
[Proceso 0] He recibido del proceso 0, el mensaje de longitud 480000, con tiempo: 0.002487 s.
[Proceso 2] He recibido del proceso 0, el mensaje de longitud 480000, con tiempo: 0.002710 s.

C:\Users\remoto\source\repos\MPI\practica7_1\x64\Release>mpiexec -n 8 practica7_1.exe
[Proceso 5] He recibido del proceso 0, el mensaje de longitud 480000, con tiempo: 0.003058 s.
[Proceso 4] He recibido del proceso 0, el mensaje de longitud 480000, con tiempo: 0.003098 s.
[Proceso 1] He recibido del proceso 0, el mensaje de longitud 480000, con tiempo: 0.002654 s.
[Proceso 3] He recibido del proceso 0, el mensaje de longitud 480000, con tiempo: 0.002939 s.
[Proceso 7] He recibido del proceso 0, el mensaje de longitud 480000, con tiempo: 0.003313 s.
[Proceso 6] He recibido del proceso 0, el mensaje de longitud 480000, con tiempo: 0.003274 s.
[Proceso 2] He recibido del proceso 0, el mensaje de longitud 480000, con tiempo: 0.002944 s.
[Proceso 0] He recibido del proceso 0, el mensaje de longitud 480000, con tiempo: 0.002622 s.
```

Figura 8. Tiempos obtenidos con "Broadcast".

```
C:\Users\remoto\source\repos\MPI\practica7_1\x64\Release>mpiexec -n 2 practica7_1.exe
[Proceso 0] He recibido del proceso 0, el mensaje de longitud 0, con tiempo: 0.001998 s.
[Proceso 1] He recibido del proceso 0, el mensaje de longitud 480000, con tiempo: 0.002518 s.

C:\Users\remoto\source\repos\MPI\practica7_1\x64\Release>mpiexec -n 4 practica7_1.exe
[Proceso 0] He recibido del proceso 0, el mensaje de longitud 0, con tiempo: 0.002561 s.
[Proceso 2] He recibido del proceso 0, el mensaje de longitud 480000, con tiempo: 0.003326 s.
[Proceso 1] He recibido del proceso 0, el mensaje de longitud 480000, con tiempo: 0.003294 s.
[Proceso 3] He recibido del proceso 0, el mensaje de longitud 480000, con tiempo: 0.003362 s.

C:\Users\remoto\source\repos\MPI\practica7_1\x64\Release>mpiexec -n 8 practica7_1.exe
[Proceso 7] He recibido del proceso 0, el mensaje de longitud 480000, con tiempo: 0.003350 s.
[Proceso 6] He recibido del proceso 0, el mensaje de longitud 480000, con tiempo: 0.003261 s.
[Proceso 3] He recibido del proceso 0, el mensaje de longitud 480000, con tiempo: 0.003222 s.
[Proceso 1] He recibido del proceso 0, el mensaje de longitud 480000, con tiempo: 0.003230 s.
[Proceso 4] He recibido del proceso 0, el mensaje de longitud 480000, con tiempo: 0.003258 s.
[Proceso 0] He recibido del proceso 0, el mensaje de longitud 0, con tiempo: 0.002839 s.
[Proceso 5] He recibido del proceso 0, el mensaje de longitud 480000, con tiempo: 0.003236 s.
[Proceso 2] He recibido del proceso 0, el mensaje de longitud 480000, con tiempo: 0.003265 s.
```

Figura 9. Tiempos obtenidos con "Punto a punto".

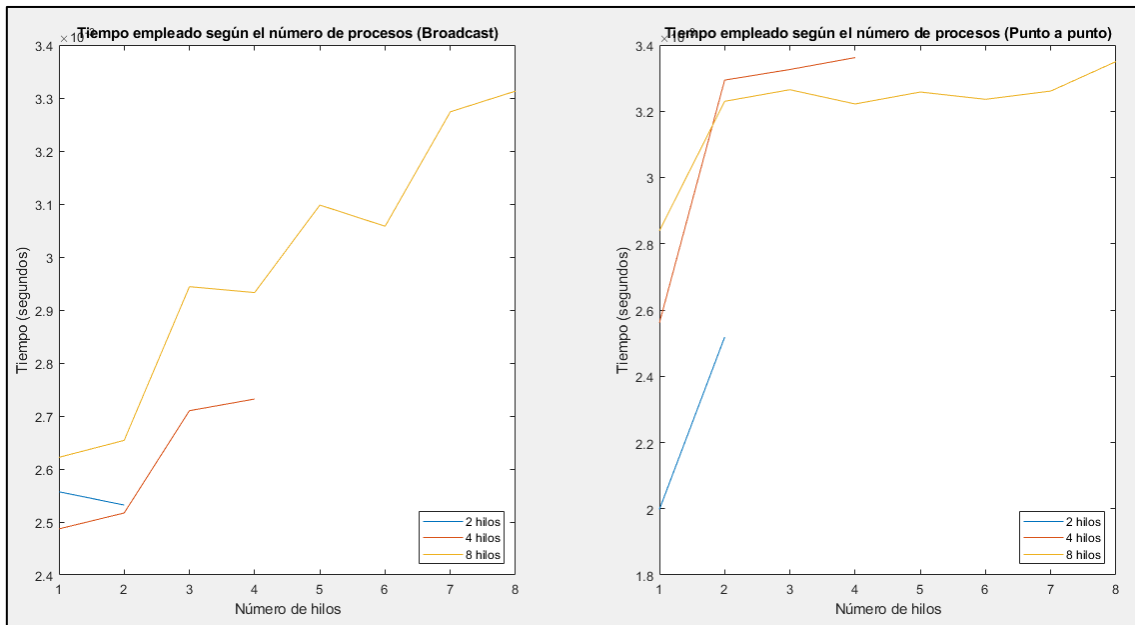


Figura 10. Comparativa de los tiempos

Podemos observar para este caso de ejecución, que los tiempos a partir de 2 hilos son mayores para el envío punto a punto frente al envío de mensajes mediante *broadcast*.

Actividad práctica 7

Se escribe un programa en el que el proceso de rango cero, que actuará como proceso principal, inicialice un vector de números en coma flotante, calcule la suma de sus elementos y la muestre por pantalla.

Tras esto, se ejecutará una operación de dispersión (*scatter*) dividiendo el vector en partes iguales, de forma que cada proceso reciba una de ellas:

- Hay tantas partes como procesos
- El tamaño del vector es múltiplo del número de procesos

A continuación, cada proceso calculará la suma de los elementos de su parte del vector y la mostrará por pantalla, identificándola con su rango.

Una vez realizado el cálculo, se ejecutará una operación de recolección (*gather*) para reunir todas las sumas parciales.

El proceso principal calculará la suma total a partir de las sumas parciales recibidas y concluirá la ejecución mostrando por pantalla el valor total de la suma, debiendo coincidir con el que se mostró al principio.

```

C:\Users\remoto\source\repos\MPI\practica7_1\x64\Release>mpiexec -n 2 practica7_1.exe
[Proceso 1] La suma de mis elementos es: 5.000000
0.00 1.00 2.00 3.00 ----La suma total es: 6.000000
[Proceso 0] La suma de mis elementos es: 1.000000
La suma de los elementos es: 6.000000

C:\Users\remoto\source\repos\MPI\practica7_1\x64\Release>mpiexec -n 4 practica7_1.exe
0.00 1.00 2.00 3.00 4.00 5.00 6.00 7.00 ----La suma total es: 28.000000
[Proceso 0] La suma de mis elementos es: 1.000000
La suma de los elementos es: 28.000000
[Proceso 2] La suma de mis elementos es: 9.000000
[Proceso 1] La suma de mis elementos es: 5.000000
[Proceso 3] La suma de mis elementos es: 13.000000

C:\Users\remoto\source\repos\MPI\practica7_1\x64\Release>mpiexec -n 8 practica7_1.exe
[Proceso 1] La suma de mis elementos es: 5.000000
0.00 1.00 2.00 3.00 4.00 5.00 6.00 7.00 8.00 9.00 10.00 11.00 12.00 13.00 14.00 15.00 --
--La suma total es: 120.000000
[Proceso 0] La suma de mis elementos es: 1.000000
La suma de los elementos es: 120.000000
[Proceso 7] La suma de mis elementos es: 29.000000
[Proceso 2] La suma de mis elementos es: 9.000000
[Proceso 6] La suma de mis elementos es: 25.000000
[Proceso 3] La suma de mis elementos es: 13.000000
[Proceso 4] La suma de mis elementos es: 17.000000
[Proceso 5] La suma de mis elementos es: 21.000000

```

Figura 11. Ejecución de varios hilos, comprobando que las sumas parciales y total coinciden.

Observamos que se ha realizado las sumas correctamente obteniendo los resultados esperados.

Actividad práctica 8

Se modifica el programa anterior para calcular cuánto tiempo tarda en calcularse la suma en serie y en paralelo. Se realizan pruebas variando el número de procesos y el tamaño del vector.

Se ha realizado pruebas en dos secciones, tomando en cuenta el número de procesos máximos que el procesador puede ejecutar simultáneamente (8). Pues, ejecutando el programa con un número de procesos menor o igual que ocho, se puede comprobar que los tiempos en paralelo se reducen según aumenta el número de procesos (figura 12).

```

C:\Users\remoto\source\repos\MPI\practica7_1\x64\Release>mpiexec -n 2 practica7_1.exe
La suma total en secuencial es: 2047996800000000.000000
El tiempo total en secuencial es: 5.590126
La suma de los elementos en paralelo es: 2047996800000000.000000
El tiempo total en paralelo es: 2.797797

C:\Users\remoto\source\repos\MPI\practica7_1\x64\Release>mpiexec -n 4 practica7_1.exe
La suma total en secuencial es: 2047996800000000.000000
El tiempo total en secuencial es: 5.568544
La suma de los elementos en paralelo es: 2047996800000000.000000
El tiempo total en paralelo es: 1.432241

C:\Users\remoto\source\repos\MPI\practica7_1\x64\Release>mpiexec -n 8 practica7_1.exe
La suma total en secuencial es: 2047996800000000.000000
El tiempo total en secuencial es: 5.587885
La suma de los elementos en paralelo es: 2047996800000000.000000
El tiempo total en paralelo es: 0.715802

```

Figura 12. Ejecución hasta 8 procesos (máxima simultaneidad), comprobando la reducción de tiempos.

Y, por otro lado, se ejecuta para un número de procesos mayor a ocho, comprobando que los tiempos de ejecución comienzan a ascender paulatinamente.

```
C:\Users\remoto\source\repos\MPI\practica7_1\x64\Release>mpiexec -n 32 practica7_1.exe
La suma total en secuencial es: 2047996800000000.000000
El tiempo total en secuencial es: 5.589090
La suma de los elementos en paralelo es: 2047996800000000.000000
El tiempo total en paralelo es: 0.804265

C:\Users\remoto\source\repos\MPI\practica7_1\x64\Release>mpiexec -n 64 practica7_1.exe
La suma total en secuencial es: 2047996800000000.000000
El tiempo total en secuencial es: 5.598831
La suma de los elementos en paralelo es: 2047996800000000.000000
El tiempo total en paralelo es: 0.822372

C:\Users\remoto\source\repos\MPI\practica7_1\x64\Release>mpiexec -n 128 practica7_1.exe
La suma total en secuencial es: 2047996800000000.000000
El tiempo total en secuencial es: 5.638705
La suma de los elementos en paralelo es: 2047996800000000.000000
El tiempo total en paralelo es: 0.920170

C:\Users\remoto\source\repos\MPI\practica7_1\x64\Release>mpiexec -n 256 practica7_1.exe
La suma total en secuencial es: 2047996800000000.000000
El tiempo total en secuencial es: 5.606319
La suma de los elementos en paralelo es: 2047996800000000.000000
El tiempo total en paralelo es: 1.458195
```

Figura 13. Ejecución de más de 8 procesos, comprobando que empieza a incrementar.

Para analizar mejor los tiempos obtenidos se realiza una comparativa mediante gráficas en MATLAB:

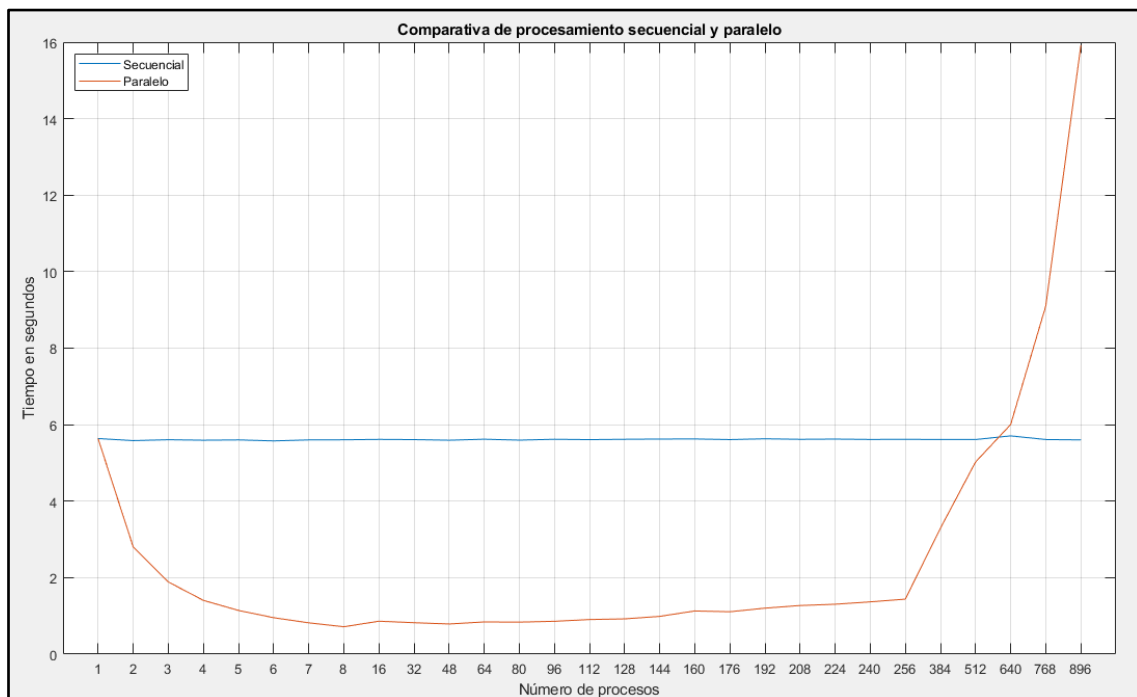


Figura 14. Comparativa entre secuencial y paralelo usando "Scatter" y "Gather".

Tal como se puede apreciar en la figura 14, el comportamiento del modo paralelo se asemeja con el de una parábola, en cambio, el secuencial es siempre constante. El modo paralelo tiene este comportamiento debido a que, únicamente, es capaz de realizar 8 procesos simultáneamente, a partir de entonces, si se añaden más procesos, este modo comienza a requerir más tiempo ya que se genera un cuello de botella entre los procesos, llegando a un punto en el que, inclusive, requiere más tiempo que el modo secuencial (entre 512 y 640 procesos el punto de corte).

Actividad práctica 9

Se escribe un programa en el que el proceso de rango cero, que actuará como proceso principal, inicialice un vector de números en coma flotante, calcule la suma de sus elementos y la muestre por pantalla.

Tras esto, el proceso principal divide el vector en partes iguales y envía cada parte a uno de los otros procesos:

- Hay tantas partes como procesos
- El proceso principal también realizar su parte del trabajo
- El tamaño del vector es múltiplo del número de procesos

Cada proceso calculará la suma de los elementos de su parte del vector y la mostrará por pantalla, identificándola con su rango.

Una vez calculada su parte, cada proceso ejecutará una operación de reducción (*MPI_SUM*) para calcular el resultado final.

El proceso principal terminará mostrando por pantalla el valor total de la suma calculada por los procesos, debiendo coincidir con el que se mostró al principio.

```
C:\Users\remoto\source\repos\MPI\practica7_1\x64\Release>mpiexec -n 2 practica7_1.exe
0.00 1.00 2.00 3.00 ----La suma total es: 6.000000
[Proceso 0] La suma de mis elementos es: 1.000000
La suma de los elementos en paralelo es: 6.000000
[Proceso 1] La suma de mis elementos es: 5.000000

C:\Users\remoto\source\repos\MPI\practica7_1\x64\Release>mpiexec -n 4 practica7_1.exe
[Proceso 2] La suma de mis elementos es: 9.000000
0.00 1.00 2.00 3.00 4.00 5.00 6.00 7.00 ----La suma total es: 28.000000
[Proceso 0] La suma de mis elementos es: 1.000000
La suma de los elementos en paralelo es: 28.000000
[Proceso 1] La suma de mis elementos es: 5.000000
[Proceso 3] La suma de mis elementos es: 13.000000

C:\Users\remoto\source\repos\MPI\practica7_1\x64\Release>mpiexec -n 8 practica7_1.exe
0.00 1.00 2.00 3.00 4.00 5.00 6.00 7.00 8.00 9.00 10.00 11.00 12.00 13.00 14.00 15.00 --
--La suma total es: 120.000000
[Proceso 0] La suma de mis elementos es: 1.000000
La suma de los elementos en paralelo es: 120.000000
[Proceso 7] La suma de mis elementos es: 29.000000
[Proceso 3] La suma de mis elementos es: 13.000000
[Proceso 4] La suma de mis elementos es: 17.000000
[Proceso 2] La suma de mis elementos es: 9.000000
[Proceso 6] La suma de mis elementos es: 25.000000
[Proceso 5] La suma de mis elementos es: 21.000000
[Proceso 1] La suma de mis elementos es: 5.000000
```

Figura 15. Ejecución de varios hilos, comprobando que las sumas parciales y total coinciden.

Observamos que se ha realizado las sumas correctamente obteniendo los resultados esperados.

Actividad práctica 10

Se modifica el programa anterior para calcular cuánto tiempo tarda en calcularse la suma en serie y en paralelo. Se realizan pruebas variando el número de procesos y el tamaño del vector.

Razona los resultados obtenidos.

A continuación, se compara las conclusiones obtenidas para la versión *scatter/gather* con las obtenidas para la versión *reduce*.

Esta práctica es análoga a la 8, cambiando simplemente el método utilizado para recolectar y realizar las sumas parciales, pues, en este caso, se emplea el método “reduce”. Por tanto, se procederá de la misma manera, realizando las pruebas en dos secciones, la primera sección, con un número de procesos menor que ocho, se presenta a continuación:

```
C:\Users\remoto\source\repos\MPI\practica7_1\x64\Release>mpiexec -n 2 practica7_1.exe
La suma total en secuencial es: 2047996800000000.000000
El tiempo total en secuencial es: 5.592950
La suma de los elementos en paralelo es: 2047996800000000.000000
El tiempo total en paralelo es: 2.801054

C:\Users\remoto\source\repos\MPI\practica7_1\x64\Release>mpiexec -n 4 practica7_1.exe
La suma total en secuencial es: 2047996800000000.000000
El tiempo total en secuencial es: 5.590878
La suma de los elementos en paralelo es: 2047996800000000.000000
El tiempo total en paralelo es: 1.427875

C:\Users\remoto\source\repos\MPI\practica7_1\x64\Release>mpiexec -n 8 practica7_1.exe
La suma total en secuencial es: 2047996800000000.000000
El tiempo total en secuencial es: 5.587894
La suma de los elementos en paralelo es: 2047996800000000.000000
El tiempo total en paralelo es: 0.715844
```

Figura 16. Ejecución hasta 8 procesos (máxima simultaneidad), comprobando la reducción de tiempos.

Tal como se puede apreciar en la figura 16, sucede lo mismo que en el apartado 8, los tiempos en paralelo comienzan a reducirse mientras se añaden más procesos. A continuación, se presentan la ejecución para un número mayor de 8 procesos:

```
C:\Users\remoto\source\repos\MPI\practica7_1\x64\Release>mpiexec -n 32 practica7_1.exe
La suma total en secuencial es: 2047996800000000.000000
El tiempo total en secuencial es: 5.590330
La suma de los elementos en paralelo es: 2047996800000000.000000
El tiempo total en paralelo es: 0.790483

C:\Users\remoto\source\repos\MPI\practica7_1\x64\Release>mpiexec -n 64 practica7_1.exe
La suma total en secuencial es: 2047996800000000.000000
El tiempo total en secuencial es: 5.594295
La suma de los elementos en paralelo es: 2047996800000000.000000
El tiempo total en paralelo es: 0.802839

C:\Users\remoto\source\repos\MPI\practica7_1\x64\Release>mpiexec -n 128 practica7_1.exe
La suma total en secuencial es: 2047996800000000.000000
El tiempo total en secuencial es: 5.599289
La suma de los elementos en paralelo es: 2047996800000000.000000
El tiempo total en paralelo es: 0.891868

C:\Users\remoto\source\repos\MPI\practica7_1\x64\Release>mpiexec -n 256 practica7_1.exe
La suma total en secuencial es: 2047996800000000.000000
El tiempo total en secuencial es: 5.630675
La suma de los elementos en paralelo es: 2047996800000000.000000
El tiempo total en paralelo es: 1.467481
```

Figura 17. Ejecución de más de 8 procesos, comprobando que empieza a incrementar.

Y una vez más, tal como se puede apreciar en la figura 17, se obtiene un comportamiento análogo al del apartado ocho. Su respectiva gráfica es:

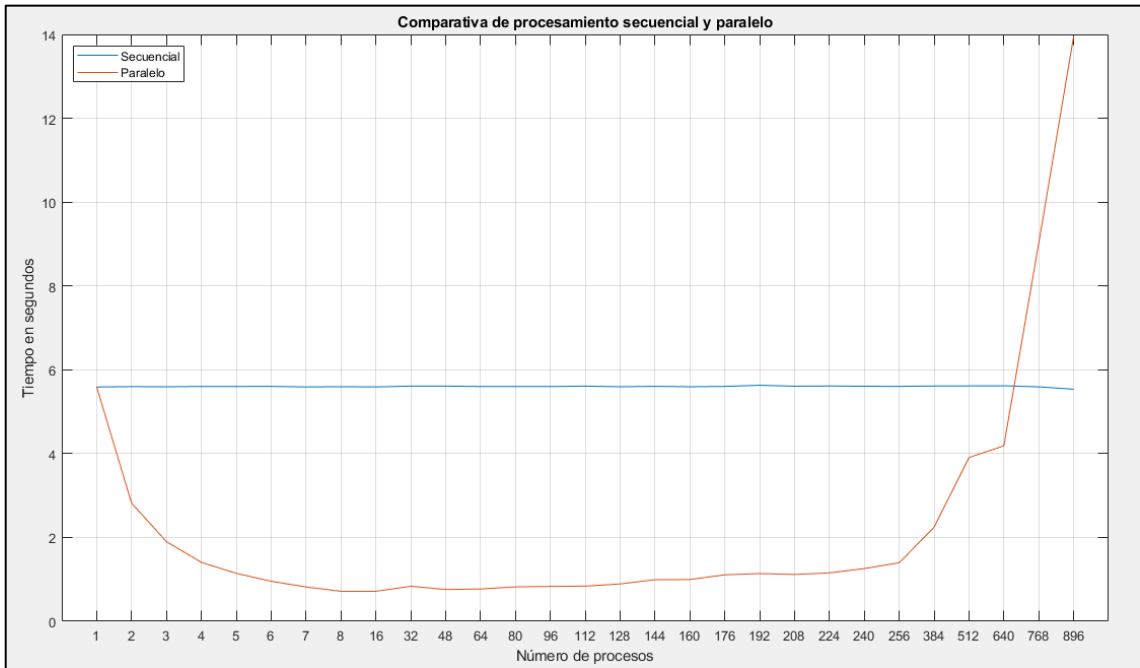


Figura 18. Comparativa entre secuencial y paralelo usando "Reduce".

Una vez presentados ambas metodologías, se procede a compararlas en una única gráfica, pues lo destacable entre este apartado y el ocho, es la utilización del método "reduce" en lugar de un "gather". Con esto se consigue mejorar los tiempos. Este comportamiento se puede apreciar en la siguiente gráfica:

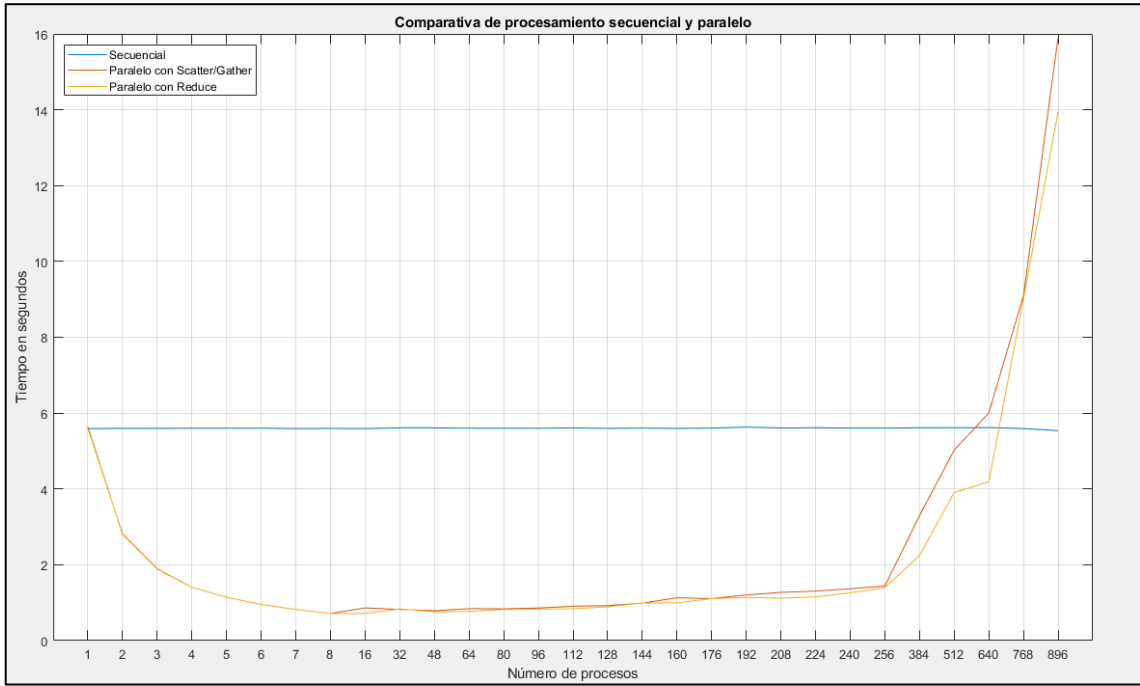


Figura 19. Comparativa usando "scatter/reduce" y "scatter/gather".