22-12-2020

Métodos Numericos para la Computación

Entrega 7

4ºCurso

Grupo Prácticas 11

Alejandro Daniel Herrera Cardenes
Carlos Eduardo Pacichana Bastidas
UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA

Indice

Practica 1

Descripción

- Ejecuta el programa cambiando el número de procesos; para ello hay que usar la línea de comandos: mpiexec.exe –n 2 miProyecto.exe
- Ejecuta el programa varias veces con un mismo número de procesos y comprueba que el resultado no es determinista, ya que el orden en el que aparecen los mensajes varía en cada ejecución

Trabajo realizado

Como podemos observar en el resultado de la ejecución ***(Resultado de ejecución)*** los mensajes no aparecen en el mismo orden en las diferentes ejecuciones debido a que no es determinista y los mensajes se generan en procesos diferentes.

MPI

Resultado Ejecución:

El resultado de la ejecución lo podemos ver en la imagen inferior (***Resultado de ejecución).***

```
C:\Users\remoto\source\repos\Entrega7\Practica1\x64\Release>mpiexec -n 4 practica1.exe
[Proceso 0] Hola desde PC-1443!
El numero total de procesos MPI es  4.
[Proceso 2] Hola desde PC-1443!
[Proceso 3] Hola desde PC-1443!
[Proceso 1] Hola desde PC-1443!

C:\Users\remoto\source\repos\Entrega7\Practica1\x64\Release>mpiexec -n 6 practica1.exe
[Proceso 3] Hola desde PC-1443!
[Proceso 1] Hola desde PC-1443!
[Proceso 2] Hola desde PC-1443!
[Proceso 0] Hola desde PC-1443!
El numero total de procesos MPI es  6.
[Proceso 4] Hola desde PC-1443!
[Proceso 5] Hola desde PC-1443!

C:\Users\remoto\source\repos\Entrega7\Practica1\x64\Release>mpiexec -n 8 practica1.exe
[Proceso 0] Hola desde PC-1443!
El numero total de procesos MPI es  8.
[Proceso 3] Hola desde PC-1443!
[Proceso 1] Hola desde PC-1443!
[Proceso 4] Hola desde PC-1443!
[Proceso 7] Hola desde PC-1443!
[Proceso 6] Hola desde PC-1443!
[Proceso 5] Hola desde PC-1443!
[Proceso 2] Hola desde PC-1443!
```

Resultado de la ejecución

Practica 2

Descripción

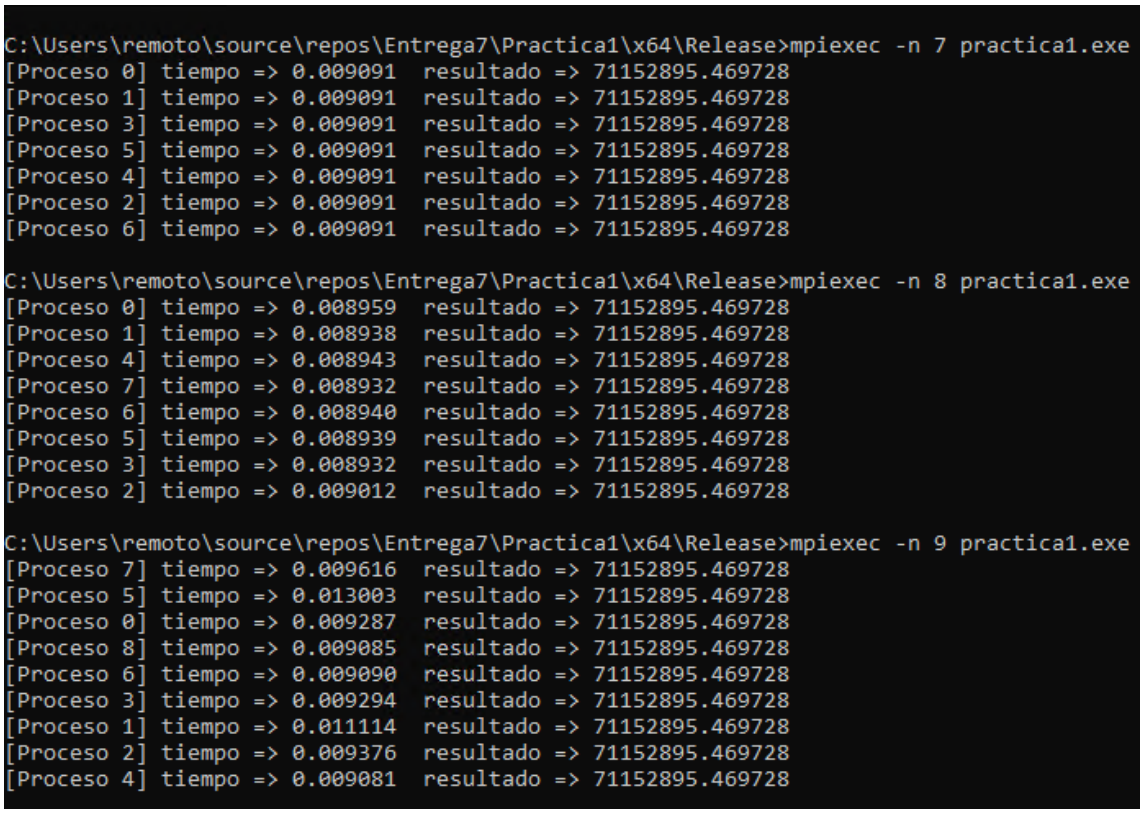
- Modifica el programa anterior para que cada proceso realice una tarea que consuma tiempo como, por ejemplo, multiplicar dos números en coma flotante varios millones de veces
- Añade al mensaje que muestra cada proceso el tiempo que ha tardado en ejecutar la operación
- Realiza varias pruebas cambiando el número total de procesos y determina si, a partir de los datos obtenidos, puedes verificar el número de procesos que es capaz de ejecutar el procesador de forma simultánea

Trabajo realizado

Se produce una varianza en los tiempos entre la ejecución de 8 a 9 procesos mientras que entre 7 y 8 no hay ningún cambio significativo por lo que podemos deducir que el número máximo de procesos simultáneos es de 8. Imprimimos el resultado debido a que si no realizamos ninguna acción con el valor que se genera de las operaciones el compilador para optimizar la ejecución omite la parte del código donde se trabaja con esto ya que al ver que no tiene uso alguno lo considera innecesario

Resultado Ejecución:

El resultado de la ejecución lo podemos ver en la imagen inferior (***Resultado de ejecución***).



Resultado de la ejecución

Practica 3

Descripción

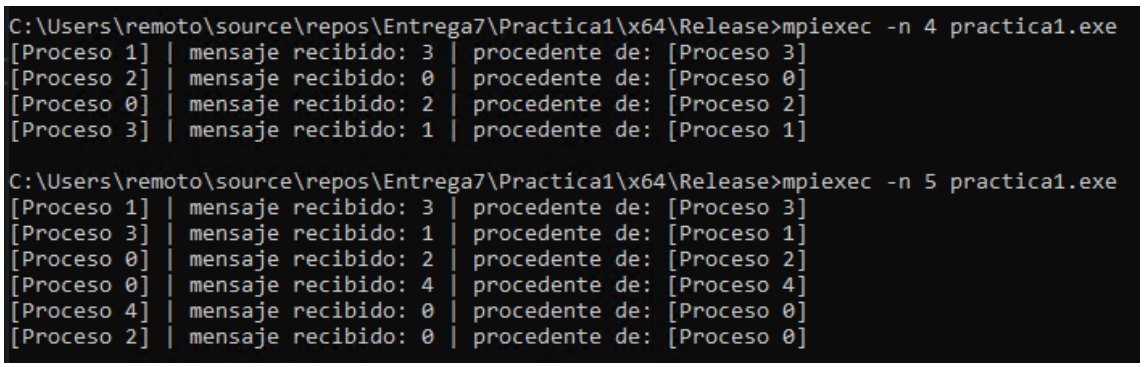
- Modifica el código anterior para que, asumiendo que el número de procesos es par, cada proceso elija otro proceso como compañero
- Tras elegir un compañero, cada proceso debe enviarle un mensaje con su identificador y recibir el mensaje recíproco que enviará su compañero
- Los procesos deben mostrar un segundo mensaje por pantalla que indique quién es su compañero y qué mensaje ha recibido de él

Trabajo realizado

Como podemos ver en los resultados cada proceso envía su identificador a otro y recibe de este el suyo, además ya que en los procesos se organizaban por parejas adaptamos el código para que en el caso de que reciba un numero de procesos impar sea el proceso 0 el que se ocupe del sobrante.

MPI

El resultado de la ejecución lo podemos ver en la imagen inferior (***Resultado de ejecución***).



Resultado de la ejecución

Practica 4

Descripción

- 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
- 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

Trabajo realizado

Modificamos los métodos de comunicación y añadimos la función ***wait*** y que como vemos por la ejecución tenemos el mismo resultado.

MPI

El resultado de la ejecución lo podemos ver en la imagen inferior (***Resultado de ejecución***).

```
C:\Users\remoto\source\repos\Entrega7\Practica1\x64\Release>mpiexec -n 6 practica1.exe
[Proceso 1] | mensaje recibido: 4 | procedente de: [Proceso 4]
[Proceso 4] | mensaje recibido: 1 | procedente de: [Proceso 1]
[Proceso 0] | mensaje recibido: 3 | procedente de: [Proceso 3]
[Proceso 3] | mensaje recibido: 0 | procedente de: [Proceso 0]
[Proceso 2] | mensaje recibido: 5 | procedente de: [Proceso 5]
[Proceso 5] | mensaje recibido: 2 | procedente de: [Proceso 2]
```

Resultado de la ejecución

Practica 5

Descripción

- El código anterior envía como mensaje un número entero, modifícalo para que envíe una cadena de caracteres con un pequeño texto
- No olvides indicar el tamaño del mensaje enviado teniendo en cuenta el carácter de final de cadena
- El texto debe ser distinto en cada proceso, por ejemplo añadiendo el identificador del proceso que lo envía (para convertir el identificador entero al carácter correspondiente súmale '0')

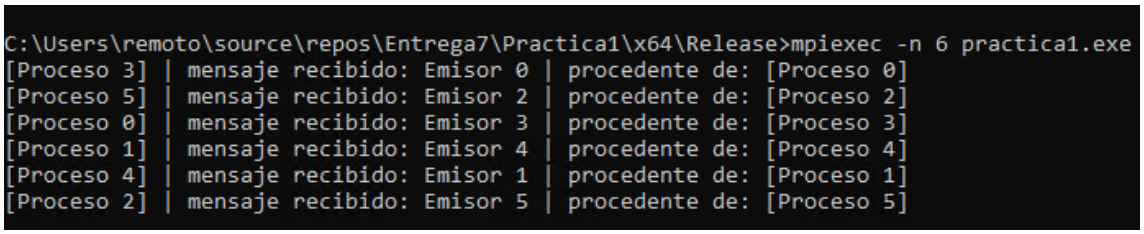
Trabajo realizado

Cambiamos el tipo de dato que se va a enviar que era un entero por un array de caracteres, modificando la variable, el tipo de dato y el tamaño en las funciones de enviar y recibir.

Para crear la cadena que se va a enviar usamos la función ***sprint_s*** para concatenar los caracteres de esta con el identificador del proceso.

MPI

El resultado de la ejecución lo podemos ver en la imagen inferior (***Resultado de ejecución***).



Resultado de la ejecución

Practica 6

Descripción

- Escribe un programa en el que el proceso de rango cero, que actuará como proceso principal, inicialice una cadena de caracteres con un determinado texto
- Tras esto, se ejecutará una operación de difusión (broadcast) para que todos los procesos reciban el mensaje; los procesos deben imprimir el mensaje por pantalla para verificar que se ha recibido correctamente
- Calcula el tiempo que tarda en realizarse la operación y compáralo con el tiempo que tarda en realizarse si se realiza con comunicaciones punto a punto; puede que necesites aumentar el tamaño del mensaje para que el tiempo empleado sea relevante

Trabajo realizado

Podemos ver los mensajes recibidos por los distintos procesos para las diferentes formas de enviarlos (punto a punto o broadcast) y el tiempo total que tardaron estos en recibirlos y que como analizamos de los datos de los tiempos la función broadcast comparada con las funciones punto a punto bloqueantes es mas rápida.

MPI

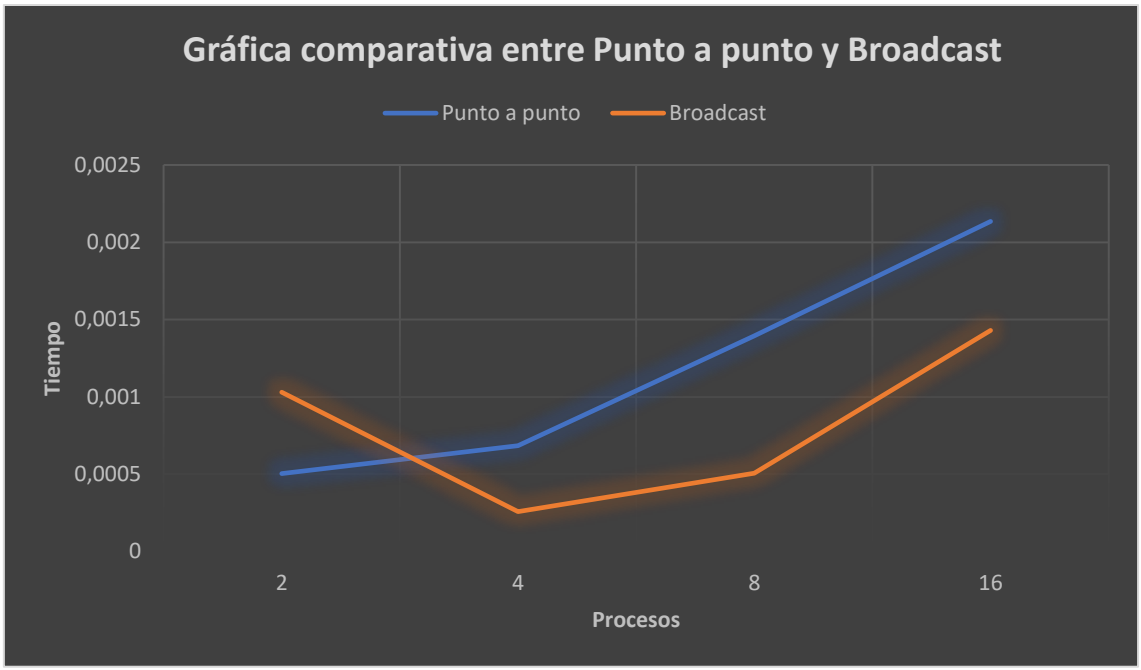
El resultado de la ejecución lo podemos ver en la imagen inferior (***Resultado de ejecución***).

```
C:\Users\remoto\source\repos\Entrega7\Practica1\x64\Release>mpiexec -n 8 practica1.exe
[Proceso 0] | Tiempo punto a punto: 0.001898
[Proceso 0] | Tiempo broadcast: 0.000518
[Proceso 1] | mensaje recibido: Texto largo para prueba... Punto a punto!
[Proceso 1] | mensaje recibido: Texto largo para prueba... Broadcast!
[Proceso 5] | mensaje recibido: Texto largo para prueba... Punto a punto!
[Proceso 5] | mensaje recibido: Texto largo para prueba... Broadcast!
[Proceso 2] | mensaje recibido: Texto largo para prueba... Punto a punto!
[Proceso 2] | mensaje recibido: Texto largo para prueba... Broadcast!
[Proceso 4] | mensaje recibido: Texto largo para prueba... Punto a punto!
[Proceso 4] | mensaje recibido: Texto largo para prueba... Broadcast!
[Proceso 6] | mensaje recibido: Texto largo para prueba... Punto a punto!
[Proceso 6] | mensaje recibido: Texto largo para prueba... Broadcast!
[Proceso 7] | mensaje recibido: Texto largo para prueba... Punto a punto!
[Proceso 7] | mensaje recibido: Texto largo para prueba... Broadcast!
[Proceso 3] | mensaje recibido: Texto largo para prueba... Punto a punto!
[Proceso 3] | mensaje recibido: Texto largo para prueba... Broadcast!
```

Resultado de la ejecución

Gráfica

Tenemos en el eje x el número de procesos y en el eje y el tiempo.



Practica 7

Descripción

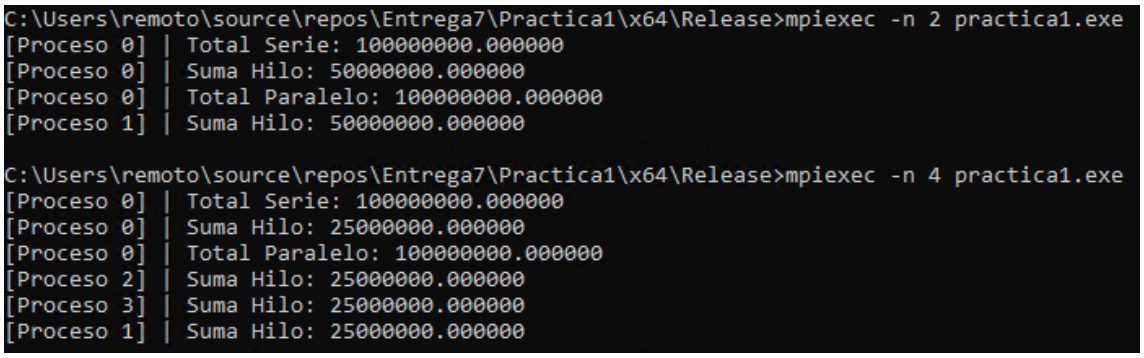
- Escribe un programa en el que el proceso de rango cero, que actuará como proceso principal, inicialice un vector de números en coma flotante, calcule la suma de sus elementos y la muestre por pantalla
- Tras esto, se ejecutará una operación de dispersión (scatter) dividiendo el vector en partes iguales, de forma que cada proceso reciba una de ellas
  - Tiene que haber tantas partes como procesos
  - El tamaño del vector debe ser 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

Trabajo realizado

Podemos observar que los cálculos tanto en serie como en paralelo dan el mismo resultado además de los cálculos de los que se encarga cada hilo por separado antes de devolverlos al proceso principal para el resultado en paralelo.

MPI

El resultado de la ejecución lo podemos ver en la imagen inferior (***Resultado de ejecución***).



Resultado de la ejecución

Practica 8

Descripción

- Modifica el programa anterior para calcular cuánto tiempo tarda en calcularse la suma en serie y en paralelo
- Haz pruebas variando el número de procesos y el tamaño del vector
- Razona los resultados obtenidos

Trabajo realizado

En esta práctica se hacía uso de los métodos ***Scatter*** y ***Gather***, en los que se reparte (Scatter) los elementos de un array a diferentes procesos para que cada uno haga operaciones con ellos. Después cada proceso realiza un Gather en el que envía al hilo principal la variable en la que se encuentra el resultado de las operaciones para que dicho hilo.

En esta practica solo imprimimos el resultado total para cada una de las maneras de calcular (Paralela y serie) debido a que utilizaremos múltiples hilos y no queremos saturar la salida del resultado para las múltiples pruebas que haremos.

De los resultados podemos sacar en claro que como vemos al doblar el número de procesos el tiempo de ejecución en paralelo se va reduciendo en aproximadamente la mitad hasta llegar al número máximo de procesos simultáneos que soporta físicamente el equipo, en este caso 8 ***(Resultado de ejecución 1).***

Al variar la longitud del vector de 10.000 a 80 podemos observar que el tiempo de ejecución en paralelo es mayor que en serie consiguiendo cada vez peores resultados según aumentamos los hilos esto es debido a que para que sea óptimo utilizar múltiples procesos el volumen de datos tiene que ser mucho mayor y al disminuir el numero de elementos del vector no podemos aprovechar la ejecución concurrente ***(Resultado de ejecución 2)***

MPI

El resultado de la ejecución lo podemos ver en la imagen inferior (***Resultado de ejecución***).

```
C:\Users\remoto\source\repos\Entrega7\Practica1\x64\Release>mpiexec -n 2 practica1.exe
---------------------------------------
Numero de elementos del Vector: 10000
---------------------------------------
[Proceso 0] | Total Serie:    100000000.000000 | Tiempo Serie:    0.089378
[Proceso 0] | Total Paralelo: 100000000.000000 | Tiempo Paralelo: 0.045259

C:\Users\remoto\source\repos\Entrega7\Practica1\x64\Release>mpiexec -n 4 practica1.exe
---------------------------------------
Numero de elementos del Vector: 10000
---------------------------------------
[Proceso 0] | Total Serie:    100000000.000000 | Tiempo Serie:    0.087839
[Proceso 0] | Total Paralelo: 100000000.000000 | Tiempo Paralelo: 0.024152

C:\Users\remoto\source\repos\Entrega7\Practica1\x64\Release>mpiexec -n 8 practica1.exe
---------------------------------------
Numero de elementos del Vector: 10000
---------------------------------------
[Proceso 0] | Total Serie:    100000000.000000 | Tiempo Serie:    0.089253
[Proceso 0] | Total Paralelo: 100000000.000000 | Tiempo Paralelo: 0.012573

C:\Users\remoto\source\repos\Entrega7\Practica1\x64\Release>mpiexec -n 16 practica1.exe
---------------------------------------
Numero de elementos del Vector: 10000
---------------------------------------
[Proceso 0] | Total Serie:    100000000.000000 | Tiempo Serie:    0.088949
[Proceso 0] | Total Paralelo: 100000000.000000 | Tiempo Paralelo: 0.015302
```

Resultado de la ejecución 1

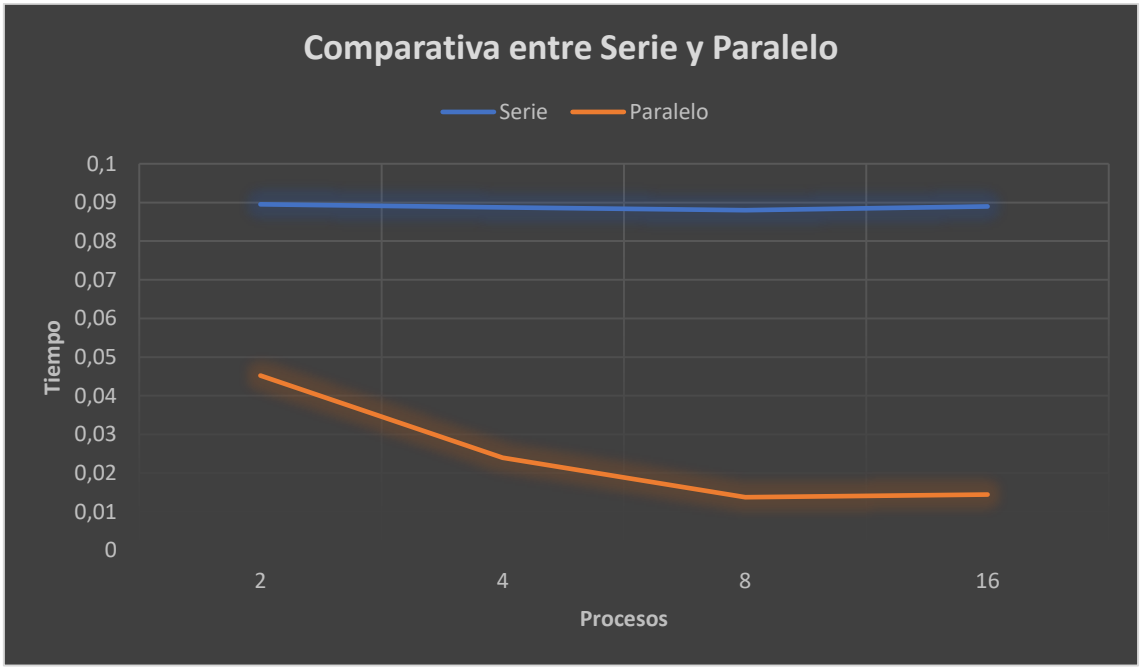
```
C:\Users\remoto\source\repos\Entrega7\Practica1\x64\Release>mpiexec -n 2 practica1.exe
---------------------------------------
Numero de elementos del Vector: 80
---------------------------------------
[Proceso 0] | Total Serie:    800000.000000 | Tiempo Serie:    0.000750
[Proceso 0] | Total Paralelo: 800000.000000 | Tiempo Paralelo: 0.000860

C:\Users\remoto\source\repos\Entrega7\Practica1\x64\Release>mpiexec -n 4 practica1.exe
---------------------------------------
Numero de elementos del Vector: 80
---------------------------------------
[Proceso 0] | Total Serie:    800000.000000 | Tiempo Serie:    0.000725
[Proceso 0] | Total Paralelo: 800000.000000 | Tiempo Paralelo: 0.001022

C:\Users\remoto\source\repos\Entrega7\Practica1\x64\Release>mpiexec -n 8 practica1.exe
---------------------------------------
Numero de elementos del Vector: 80
---------------------------------------
[Proceso 0] | Total Serie:    800000.000000 | Tiempo Serie:    0.000888
[Proceso 0] | Total Paralelo: 800000.000000 | Tiempo Paralelo: 0.001612
```

Resultado de la ejecución 2

Gráfica

Separaremos las gráficas probando con distintas cantidades de procesos usados.

Tendremos en el eje x el número de procesos y en el eje y el tiempo.



Practica 9

Descripción

- 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 debe dividir el vector en partes iguales y enviar cada parte a uno de los otros procesos
  - Tiene que haber tantas partes como procesos
  - El proceso principal también debe realizar su parte del trabajo
  - El tamaño del vector debe ser 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

Trabajo realizado

Podemos observar como el calculo total en serie y en paralelo dan el mismo resultado además de los cálculos de los que se encarga cada hilo por separado antes de devolverlos al proceso principal para el resultado en paralelo.

MPI

El resultado de la ejecución lo podemos ver en la imagen inferior (***Resultado de ejecución***).

```
C:\Users\remoto\source\repos\Entrega7\Practica1\x64\Release>mpiexec -n 2 practica1.exe
[Proceso 0] | Total Serie:    100000000.000000
[Proceso 0] | Suma Hilo: 50000000.000000
[Proceso 0] | Total Paralelo: 100000000.000000
[Proceso 1] | Suma Hilo: 50000000.000000

C:\Users\remoto\source\repos\Entrega7\Practica1\x64\Release>mpiexec -n 4 practica1.exe
[Proceso 3] | Suma Hilo: 25000000.000000
[Proceso 1] | Suma Hilo: 25000000.000000
[Proceso 0] | Total Serie:    100000000.000000
[Proceso 0] | Suma Hilo: 25000000.000000
[Proceso 0] | Total Paralelo: 100000000.000000
[Proceso 2] | Suma Hilo: 25000000.000000
```

Resultado de la ejecución

Practica 10

Descripción

- Modifica el programa anterior para calcular cuánto tiempo tarda en calcularse la suma en serie y en paralelo
- Haz pruebas variando el número de procesos y el tamaño del vector
- Razona los resultados obtenidos
- Compara las conclusiones obtenidas para la versión scatter/gather con las obtenidas para la versión reduce

Trabajo realizado

Esta última actividad es muy parecida a la anterior pero en vez de usar un gather para recoger los resultados de las sumas de cada proceso, usamos un ***reduce*** que además suma todos los valores recogidos en una misma variable, reduciendo el código para realizar la actividad, puesto que no tenemos que sumar posteriormente los valores.

MPI

El resultado de la ejecución lo podemos ver en la imagen inferior (***Resultado de ejecución***).

```
C:\Users\remoto\source\repos\Entrega7\Practica1\x64\Release>mpiexec -n 2 practica1.exe
--------------------------------------
Numero de elementos del Vector: 10000
--------------------------------------
[Proceso 0] | Total Serie:    100000000.000000 | Tiempo Serie:    0.088516
[Proceso 0] | Total Paralelo: 100000000.000000 | Tiempo Paralelo: 0.045447

C:\Users\remoto\source\repos\Entrega7\Practica1\x64\Release>mpiexec -n 4 practica1.exe
--------------------------------------
Numero de elementos del Vector: 10000
--------------------------------------
[Proceso 0] | Total Serie:    100000000.000000 | Tiempo Serie:    0.089404
[Proceso 0] | Total Paralelo: 100000000.000000 | Tiempo Paralelo: 0.025272

C:\Users\remoto\source\repos\Entrega7\Practica1\x64\Release>mpiexec -n 8 practica1.exe
--------------------------------------
Numero de elementos del Vector: 10000
--------------------------------------
[Proceso 0] | Total Serie:    100000000.000000 | Tiempo Serie:    0.089540
[Proceso 0] | Total Paralelo: 100000000.000000 | Tiempo Paralelo: 0.012687

C:\Users\remoto\source\repos\Entrega7\Practica1\x64\Release>mpiexec -n 16 practica1.exe
--------------------------------------
Numero de elementos del Vector: 10000
--------------------------------------
[Proceso 0] | Total Serie:    100000000.000000 | Tiempo Serie:    0.089193
[Proceso 0] | Total Paralelo: 100000000.000000 | Tiempo Paralelo: 0.015058
```

Resultado de la ejecución 1

```
C:\Users\remoto\source\repos\Entrega7\Practica1\x64\Release>mpiexec -n 2 practica1.exe
--------------------------------------
Numero de elementos del Vector: 80
--------------------------------------
[Proceso 0] | Total Serie:    800000.000000 | Tiempo Serie:    0.000732
[Proceso 0] | Total Paralelo: 800000.000000 | Tiempo Paralelo: 0.000938

C:\Users\remoto\source\repos\Entrega7\Practica1\x64\Release>mpiexec -n 4 practica1.exe
--------------------------------------
Numero de elementos del Vector: 80
--------------------------------------
[Proceso 0] | Total Serie:    800000.000000 | Tiempo Serie:    0.000739
[Proceso 0] | Total Paralelo: 800000.000000 | Tiempo Paralelo: 0.000719

C:\Users\remoto\source\repos\Entrega7\Practica1\x64\Release>mpiexec -n 8 practica1.exe
--------------------------------------
Numero de elementos del Vector: 80
--------------------------------------
[Proceso 0] | Total Serie:    800000.000000 | Tiempo Serie:    0.000726
[Proceso 0] | Total Paralelo: 800000.000000 | Tiempo Paralelo: 0.001443
```

Resultado de la ejecución 2

Gráfica

Separaremos las gráficas probando con distintas cantidades de procesos usados.

Tendremos en el eje x el número de procesos y en el eje y el tiempo.