MÉTODOS NUMÉRICOS PARA LA COMPUTACIÓN

**Tema 6: Matrices dispersas**

**2020/21**

21 de diciembre de 2020

**Grupo 03:** José María Amusquívar Poppe y Prashant Jeswani Tejwani

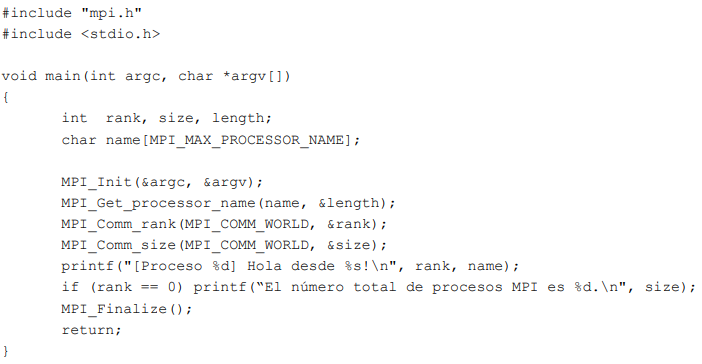
Universidad de Las Palmas de Gran Canaria

Escuela de Ingeniería en Informática

Índice

[Actividad práctica 1 2](#_Toc59453825)

# Actividad práctica 1

Este código genera procesos que muestran un mensaje por pantalla:

Ejecuta el programa cambiando el número de procesos; para ello hay que usar la línea de comandos:

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.

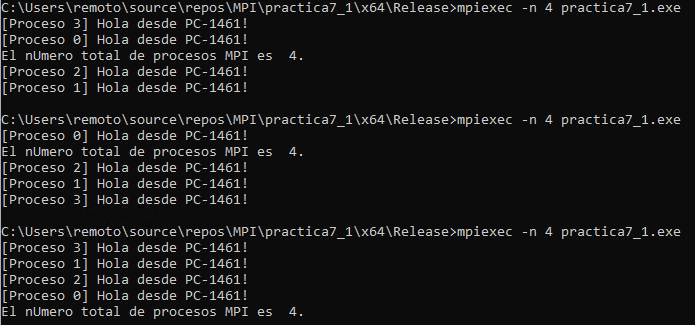
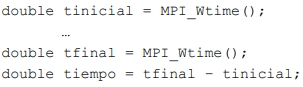


Figura 1. Ejecución empleando 4 hilos.

# Actividad práctica 2

Podemos medir el tiempo que tarda una operación:

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.

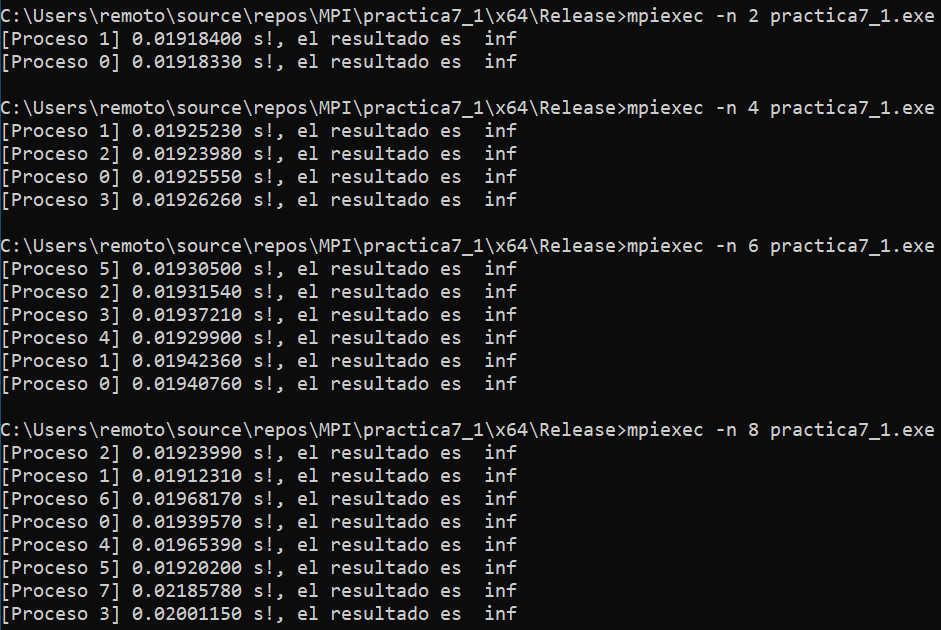
Realiza varias pruebas cambiando el número total de procesos y determina si, a partir de los datos obtenidos, puedes verificar el número de procesos que es capaz de ejecutar el procesador de forma simultánea.

Figura 2. Ejecución empleando hasta 8 hilos simultáneos.

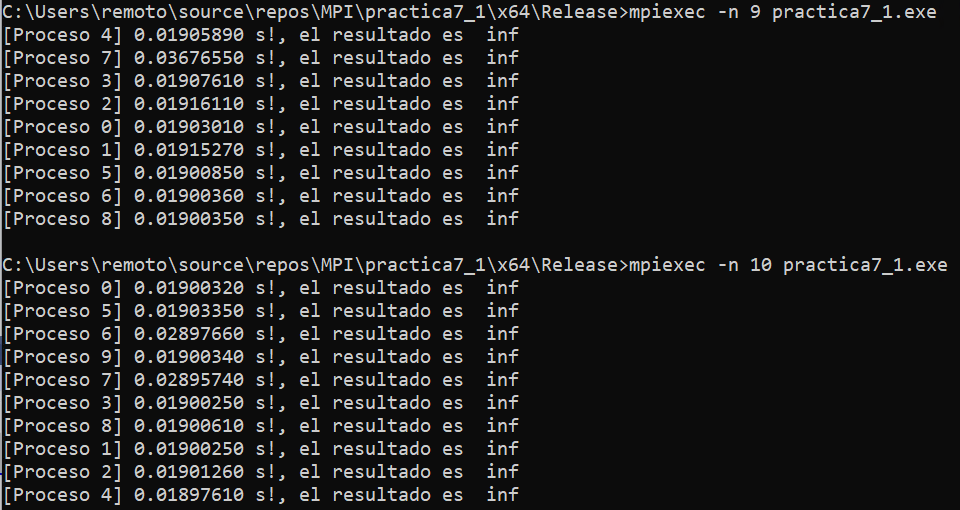


Figura 3. Ejecución empleando 9 y 10 hilos.

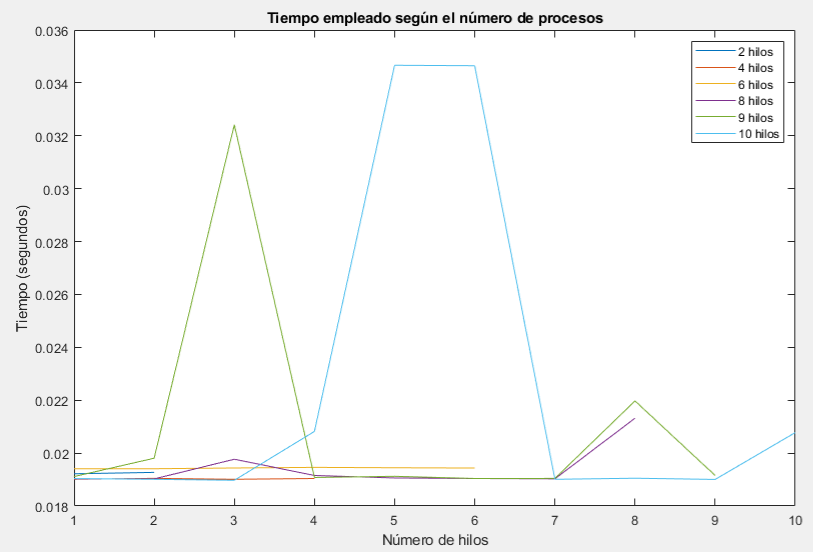
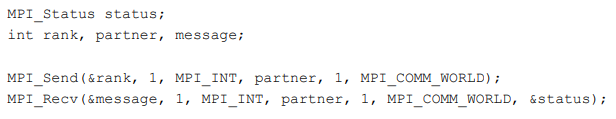
Por tanto, se comprueba que, a partir de los 9 hilos simultáneos, existen hilos que sobrepasan la media temporal de ejecución, por lo que se puede determinar que el procesador, únicamente, es capaz de ejecutar 8 hilos de manera simultánea.

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

# Actividad práctica 3

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.

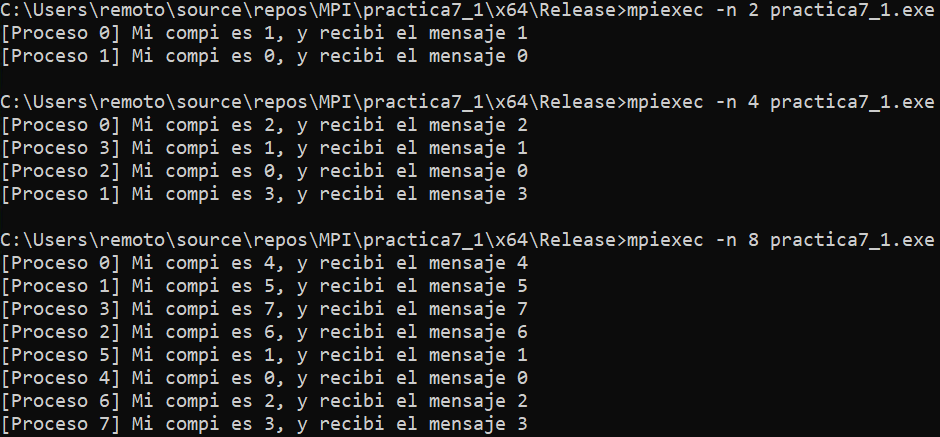
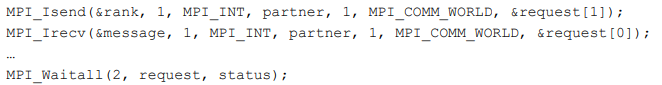


Figura 5. Ejecución con varios hilos, comprobando su funcionamiento.

# Actividad práctica 4

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:



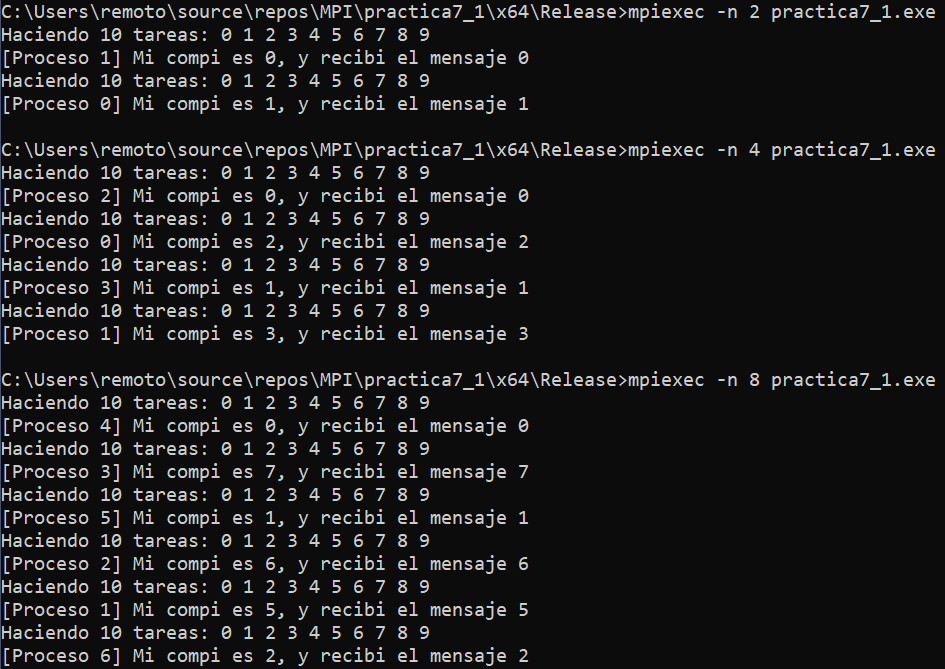


Figura 6. Ejecución empleando varios hilos, realizando una tarea en segundo plano.

# Actividad práctica 5

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’).

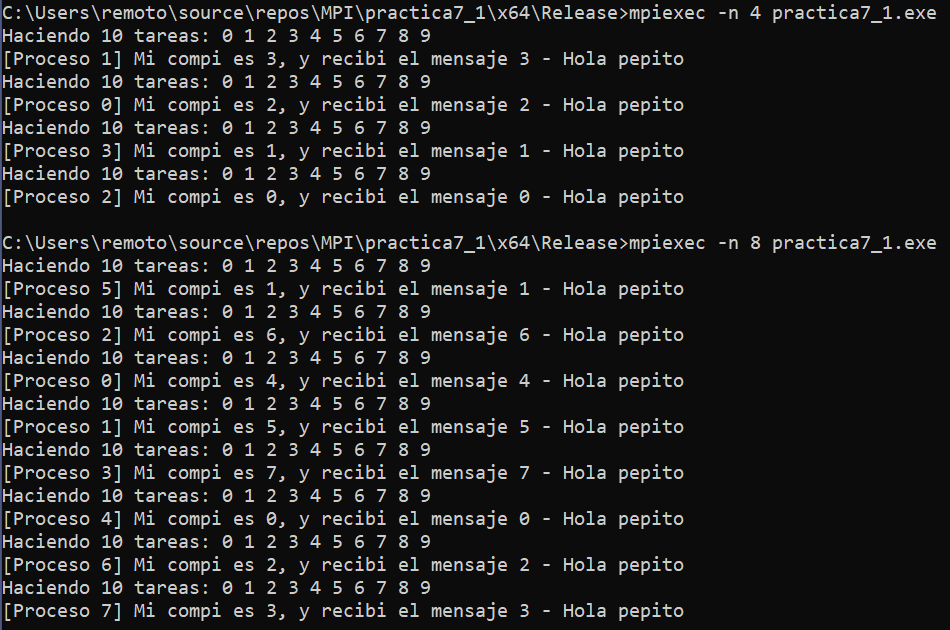


Figura 7. Ejecución de varios hilos, enviando un mensaje de texto y realizando una operación en segundo plano.

# Actividad práctica 6

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.

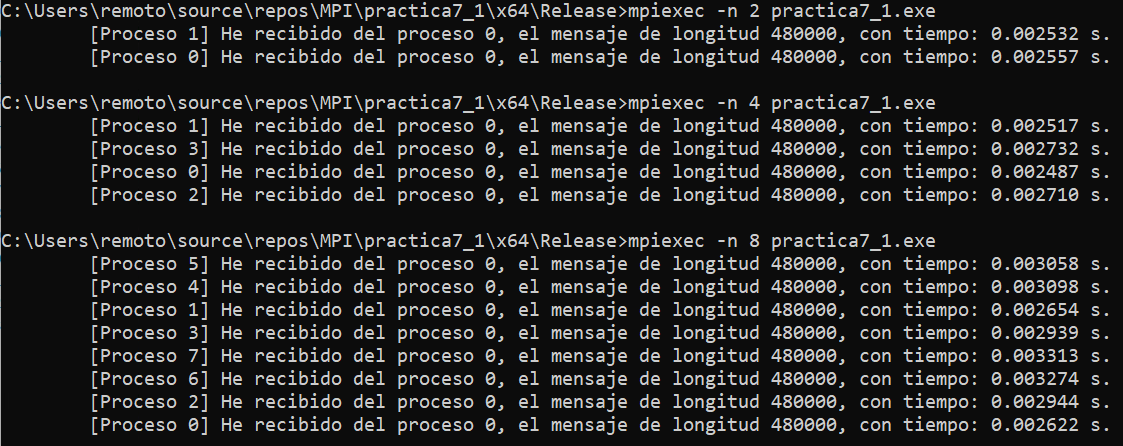
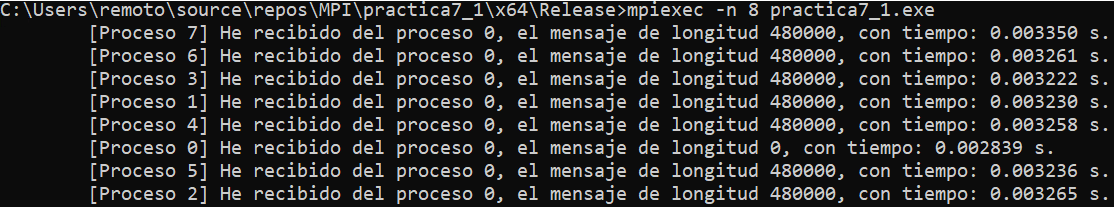
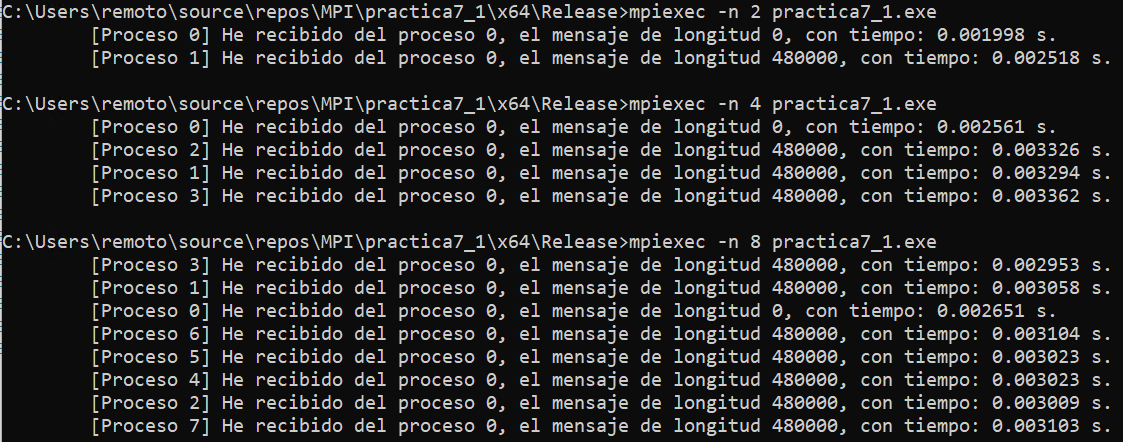


Figura 8. Tiempos obtenidos con "Broadcast".

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

# Actividad práctica 7

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.

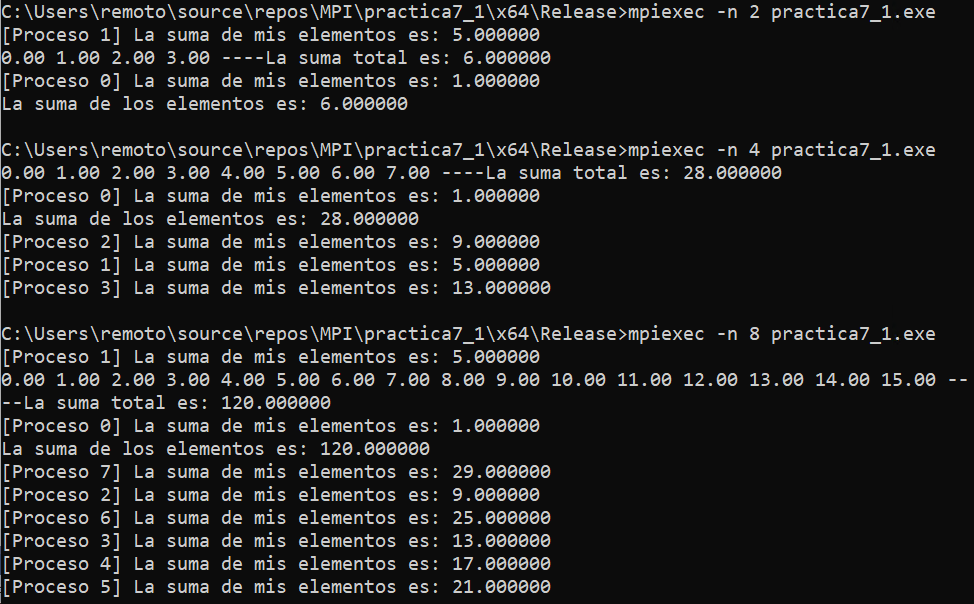


Figura 10. Ejecución de varios hilos, comprobando que las sumas parciales y total coinciden.

# Actividad práctica 8

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.

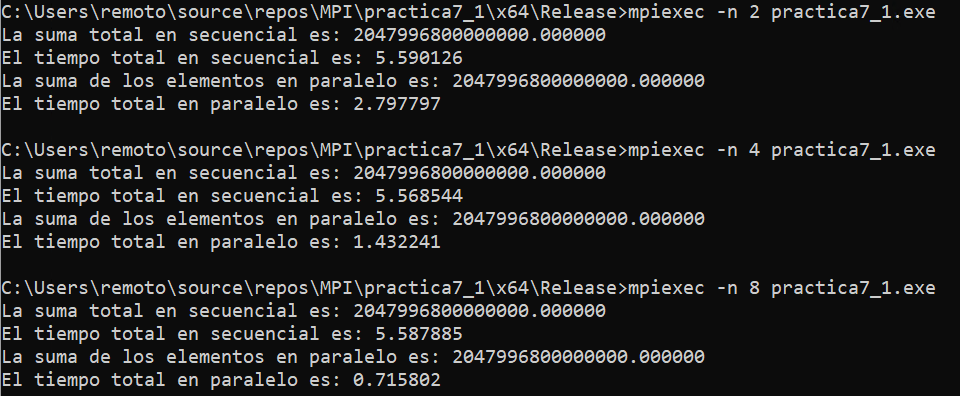
Razona los resultados obtenidos.

Figura 11. Ejecución hasta 8 procesos (máxima simultaneidad), comprobando la reducción de tiempos.

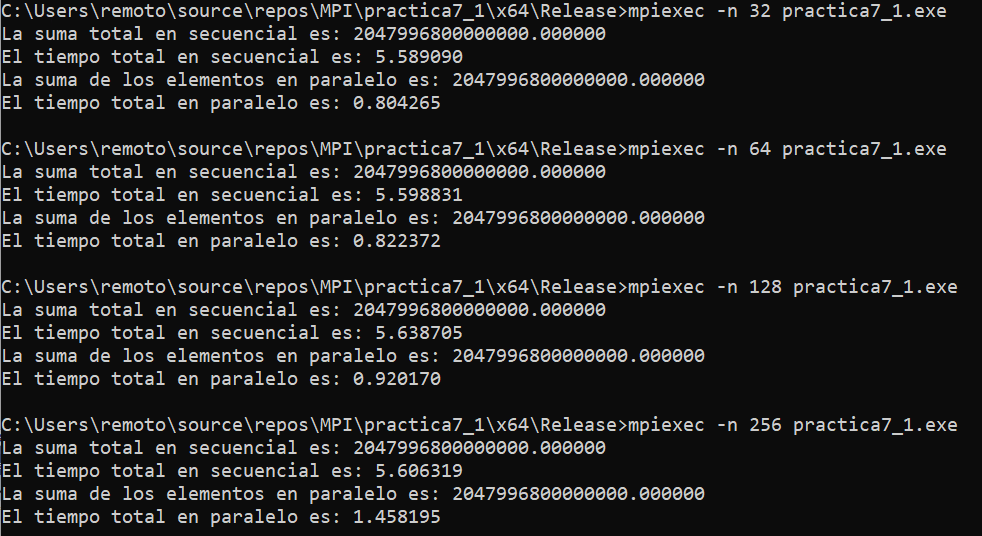


Figura 12. Ejecución de más de 8 procesos, comprobando que empieza a incrementar.

# Actividad práctica 9

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.

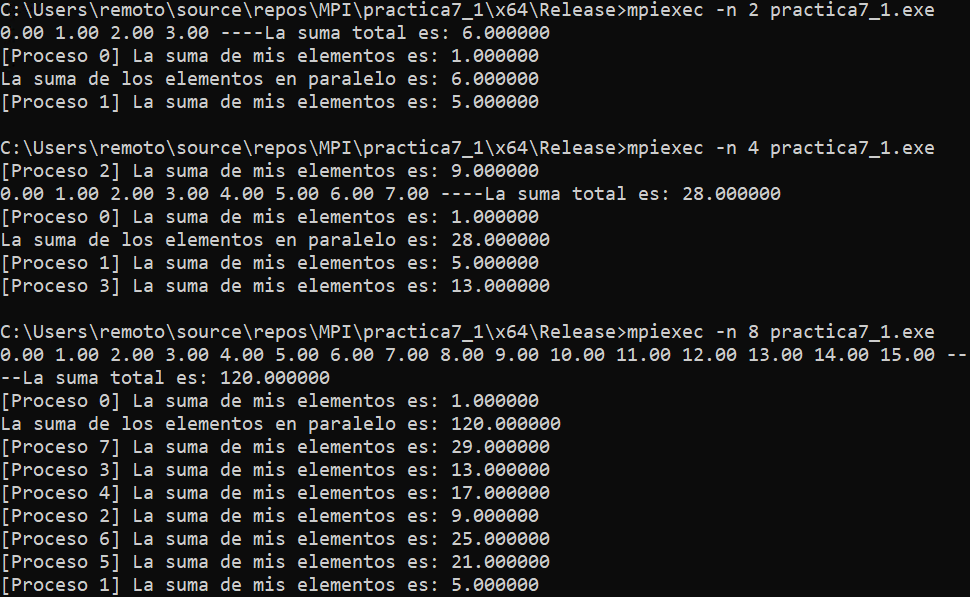


Figura 13. Ejecución de varios hilos, comprobando que las sumas parciales y total coinciden.

# Actividad práctica 10

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*.

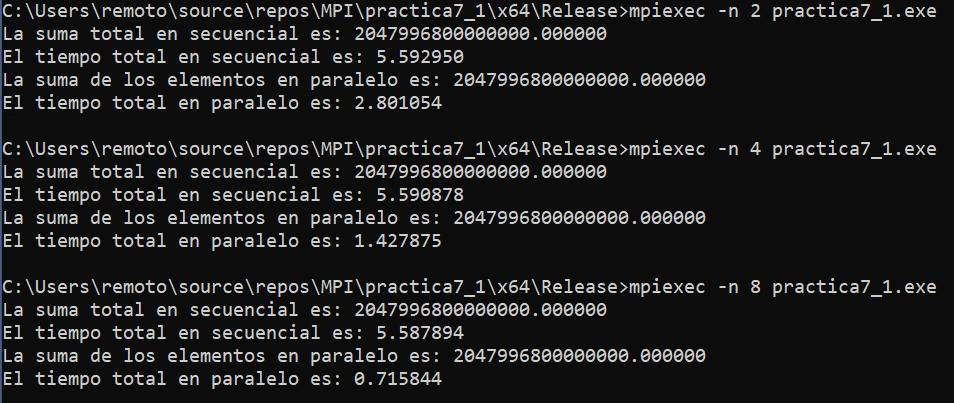


Figura 14. Ejecución hasta 8 procesos (máxima simultaneidad), comprobando la reducción de tiempos.

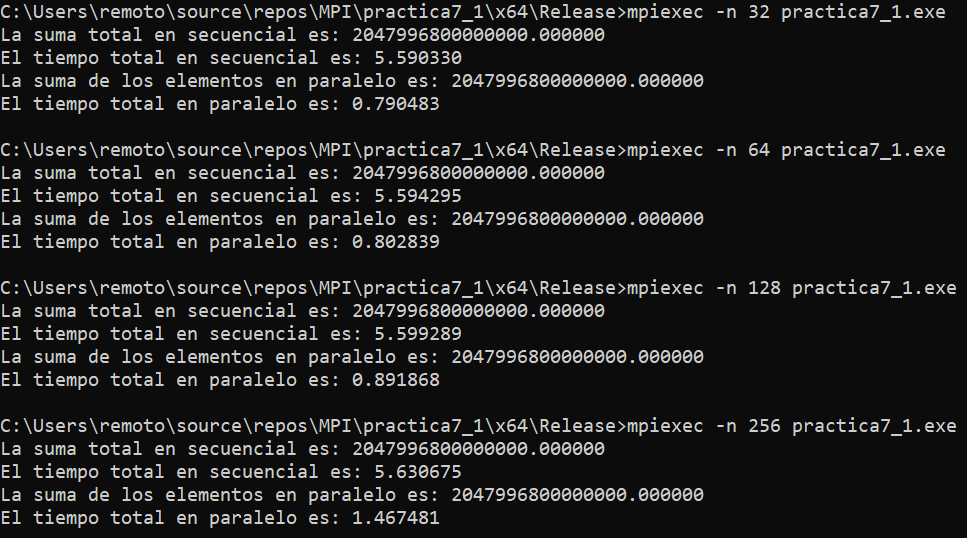


Figura 15. Ejecución de más de 8 procesos, comprobando que empieza a incrementar.