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 Tejwani

Universidad de Las Palmas de Gran Canaria

Escuela de Ingeniería en Informática

Índice

[Actividad práctica 1 3](#_Toc59471181)

[Actividad práctica 2 4](#_Toc59471182)

[Actividad práctica 3 6](#_Toc59471183)

[Actividad práctica 4 7](#_Toc59471184)

[Actividad práctica 5 8](#_Toc59471185)

[Actividad práctica 6 8](#_Toc59471186)

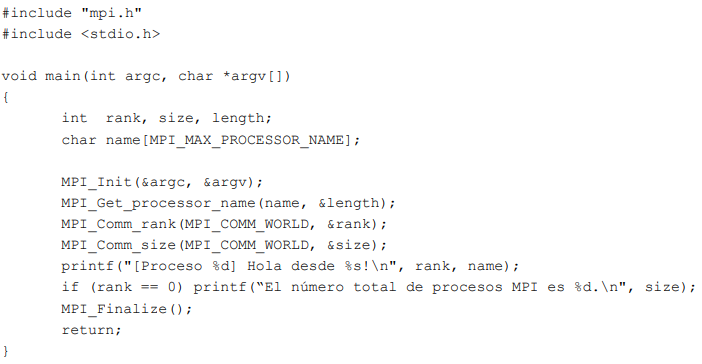
[Actividad práctica 7 9](#_Toc59471187)

[Actividad práctica 8 10](#_Toc59471188)

[Actividad práctica 9 11](#_Toc59471189)

[Actividad práctica 10 12](#_Toc59471190)

# Actividad práctica 1

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

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

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.

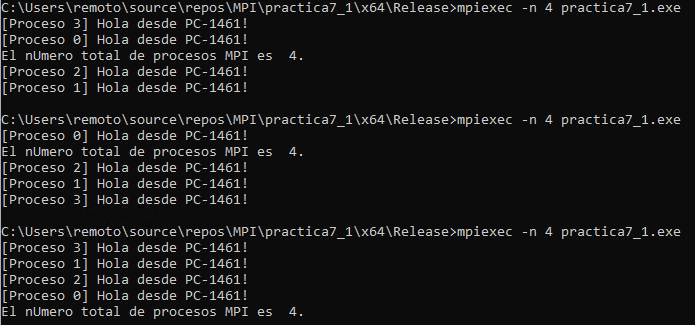
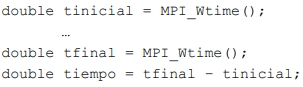


Figura . Ejecución empleando 4 hilos.

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:

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.

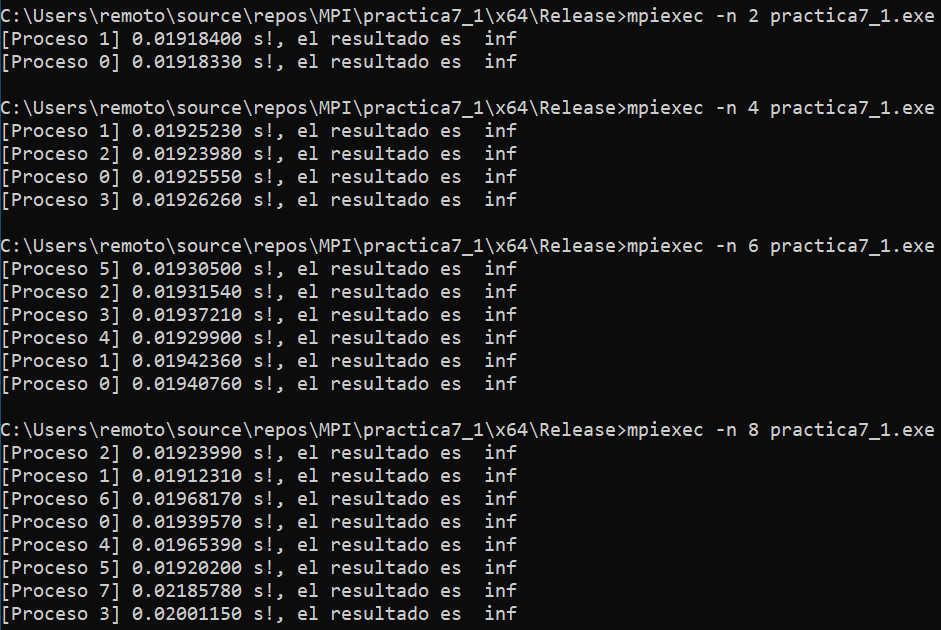
Se han realizado varias pruebas cambiando el número total de procesos y para determina si, a partir de los datos obtenidos, podemos verificar el número de procesos que es capaz de ejecutar el procesador de forma simultánea.

Figura . Ejecución empleando hasta 8 hilos simultáneos.

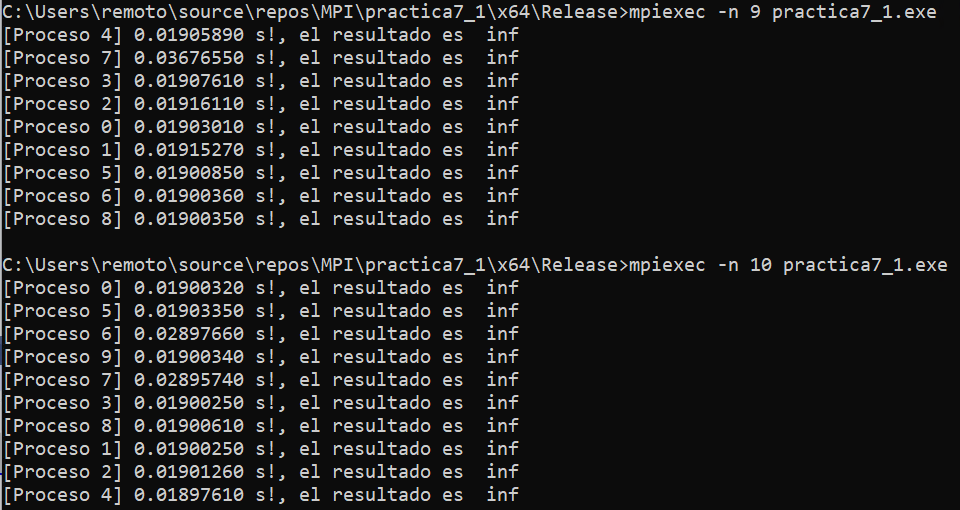


Figura . Ejecución empleando 9 y 10 hilos.

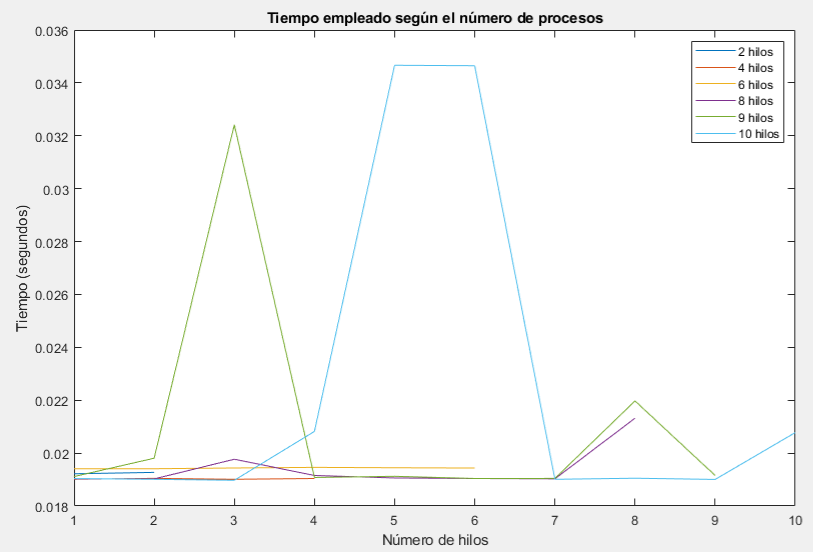
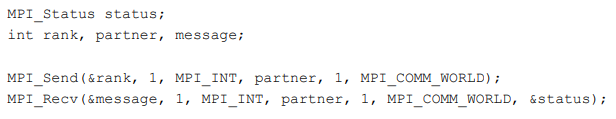
Por tanto, se comprueba que, a partir de los 9 hilos simultáneos, existen hilos que sobrepasan la media temporal de ejecución, por lo que se puede determinar que el procesador, únicamente, es capaz de ejecutar 8 hilos de manera simultánea.

Figura . 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:



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

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

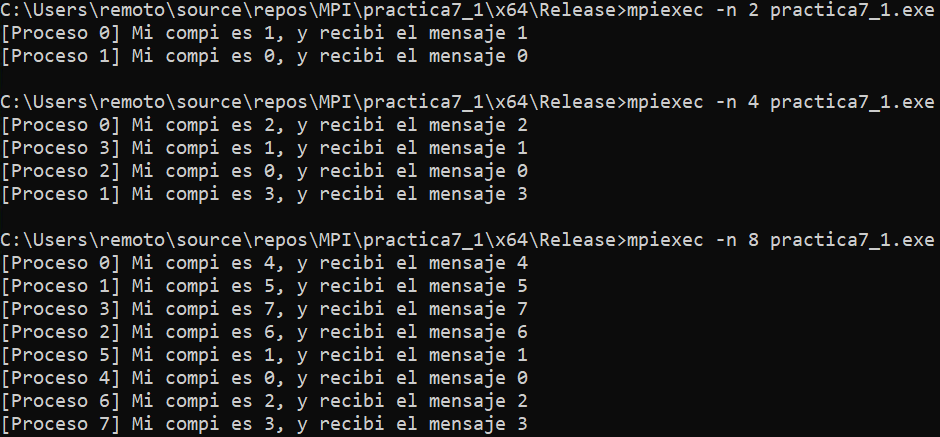
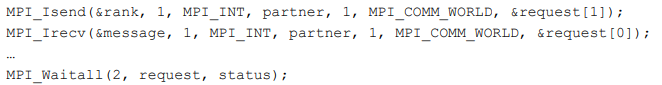
Observamos que los mensajes son enviados y recibidos correctamente para distintos número de hilos.

Figura . Ejecución con varios hilos, comprobando su funcionamiento.

# 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:

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:



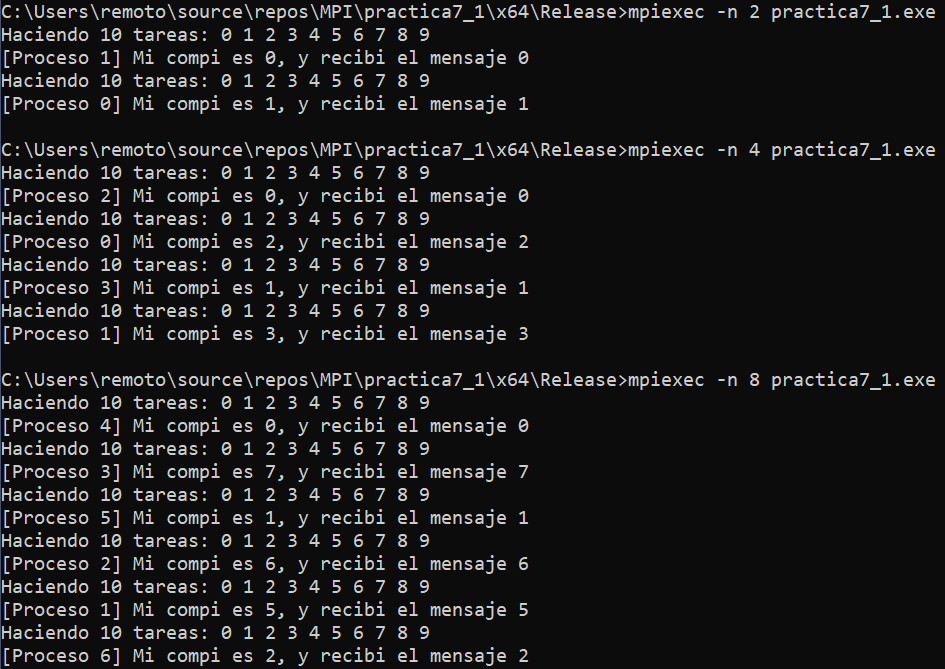


Figura . Ejecución empleando varios hilos, realizando una tarea en segundo plano.

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.

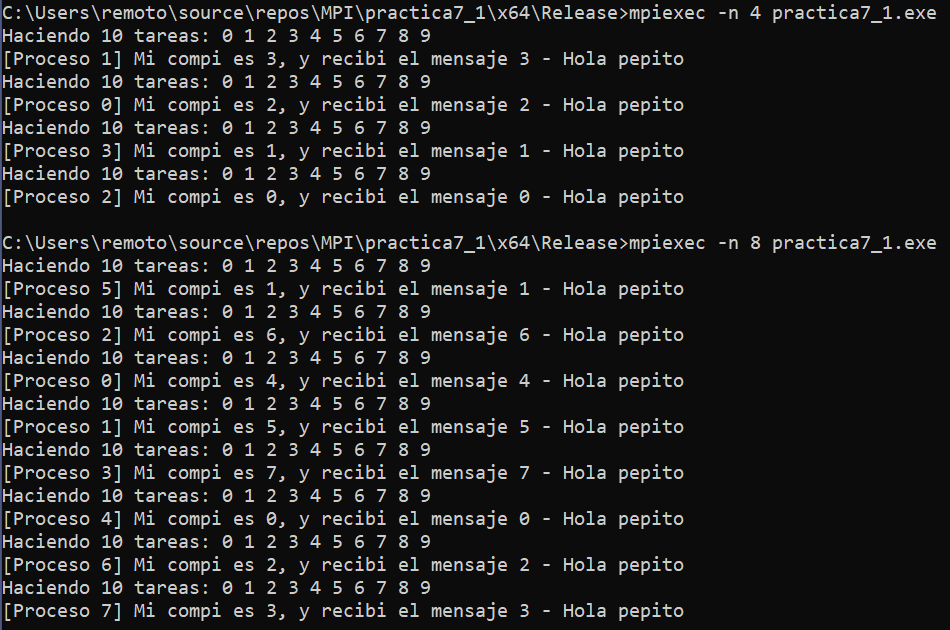
Una vez más, observamos que los mensajes se han mandado y recibido de forma correcta con un mensaje junto al identificador del emisor.

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

# 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 aumentar el tamaño del mensaje para que el tiempo empleado sea relevante.

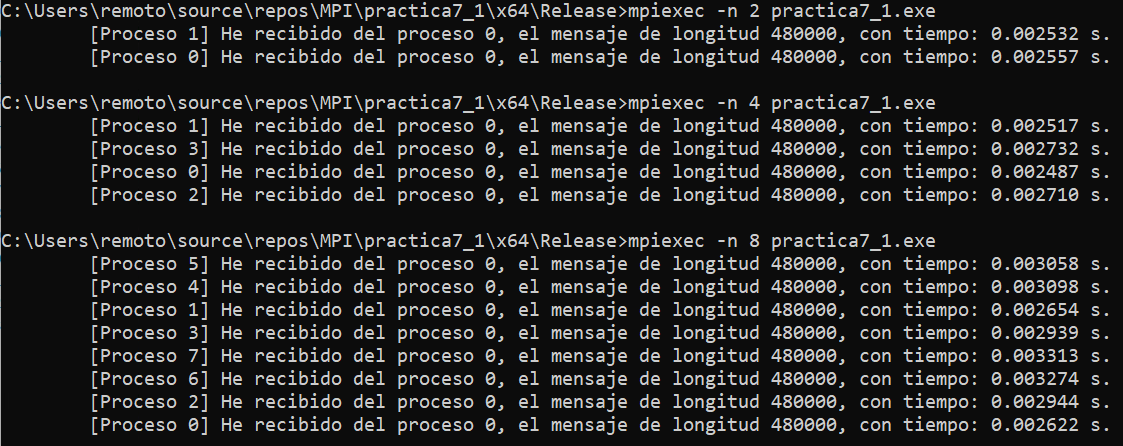
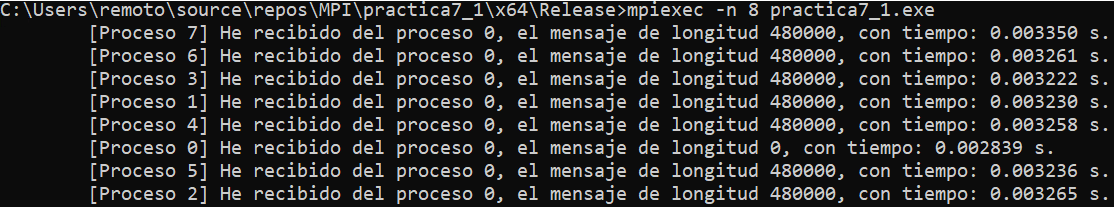
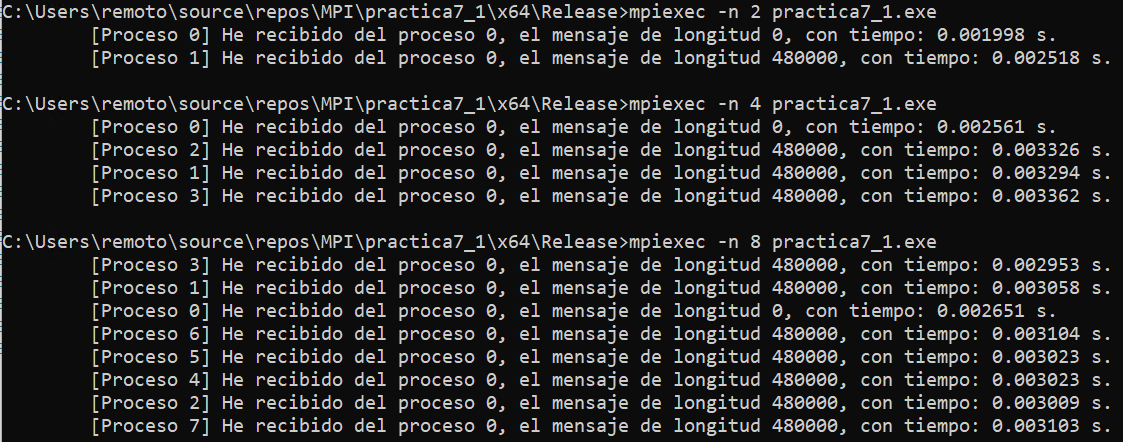


Figura . Tiempos obtenidos con "Broadcast".

Figura . Tiempos obtenidos con "Punto a punto".

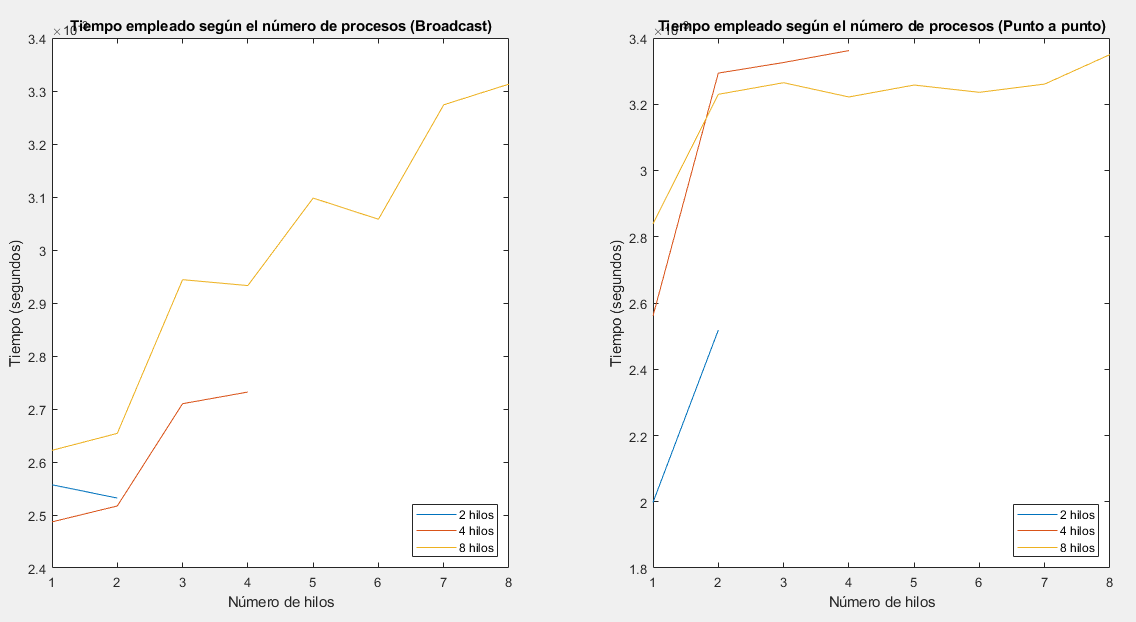


Figura . Comparativa de los tiempos

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.

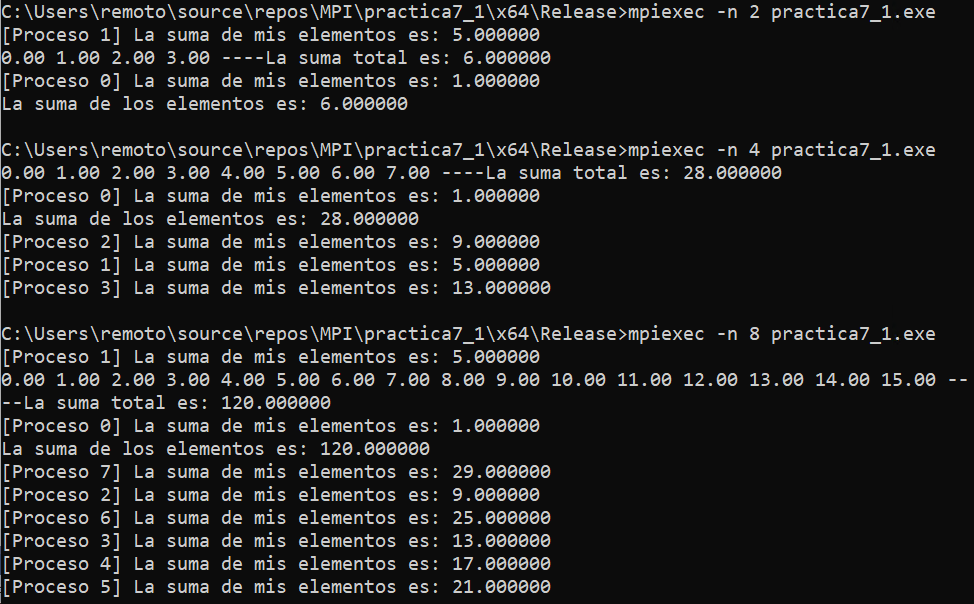
Observamos que se ha realizado las sumas correctamente obteniendo los resultados esperados.

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

# 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.

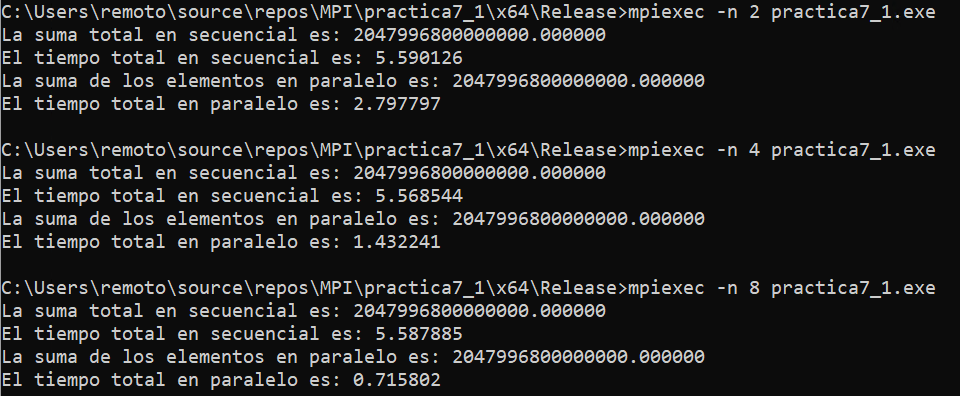


Figura . Ejecución hasta 8 procesos (máxima simultaneidad), comprobando la reducción de tiempos.

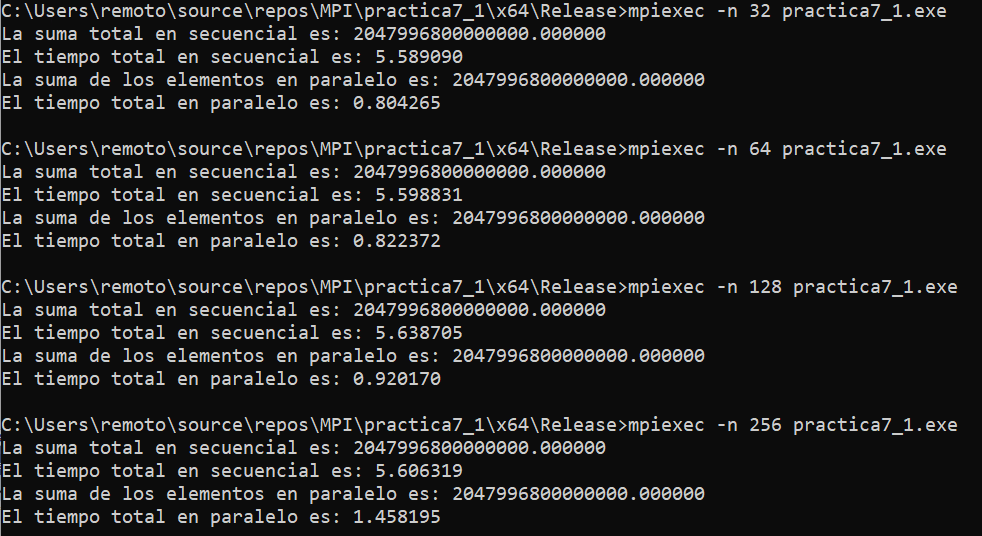
Para analizar mejor los tiempos obtenidos se realiza una comparativa mediante gráficas en MATLAB:

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

(fotito de gráfica + explicación)

# 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.

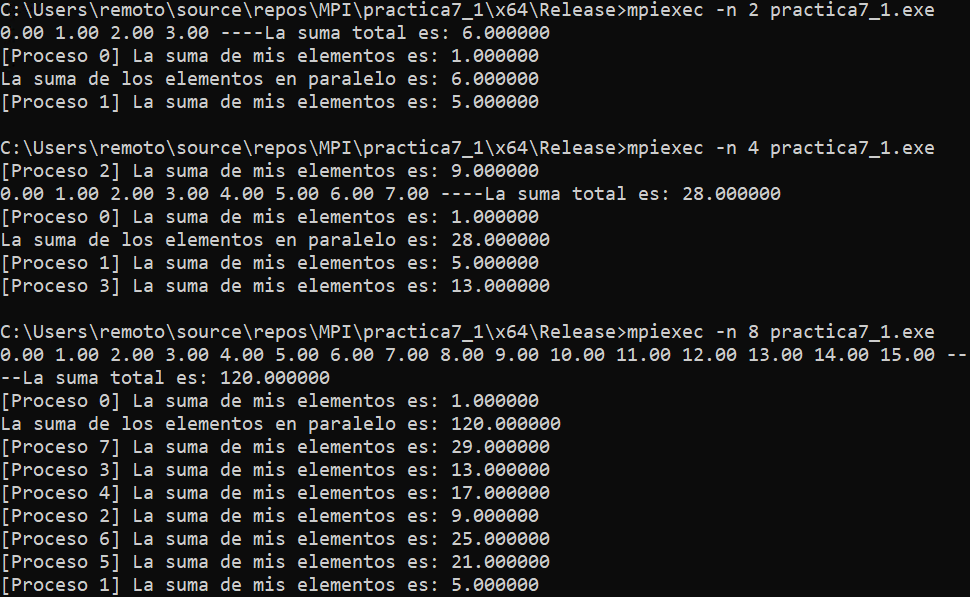
Observamos que se ha realizado las sumas correctamente obteniendo los resultados esperados.

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

# 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*.

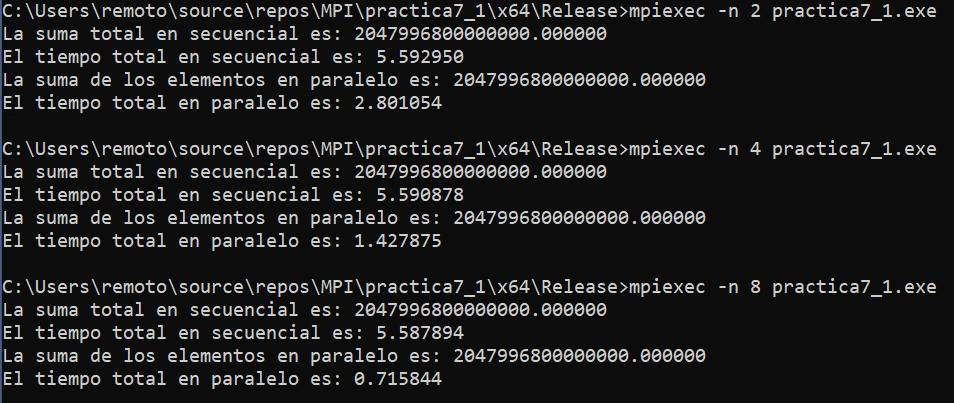


Figura . Ejecución hasta 8 procesos (máxima simultaneidad), comprobando la reducción de tiempos.

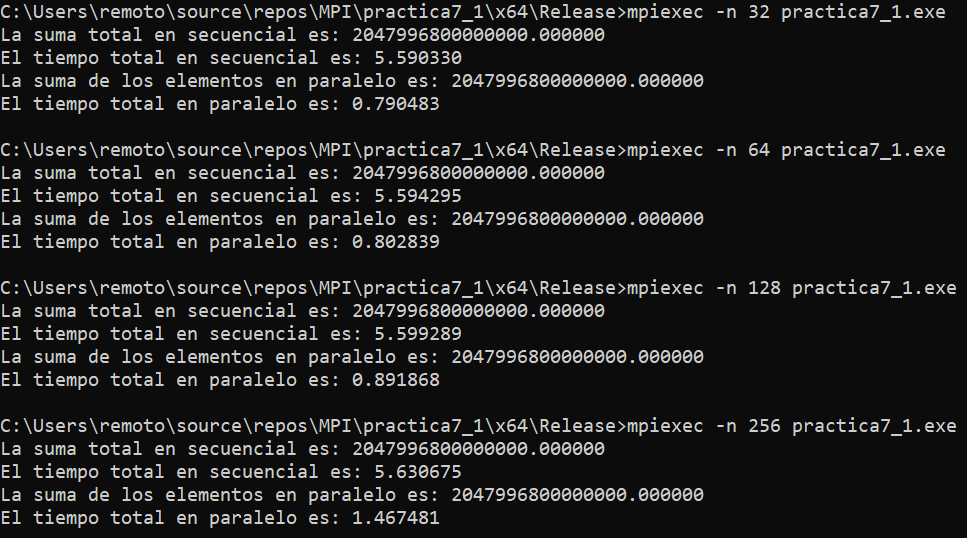


Figura . Ejecución de más de 8 procesos, comprobando que empieza a incrementar.