Práctica 5

**Optimización y Sincronización en OpenMP**

Jordi Blasco Lozano

Computación de alto rendimiento

Grado en Inteligencia Artificial

## Indice:

[Indice: 2](#_Toc190980674)

[1. INSTALACIÓN Y CONFIGURACIÓN 2](#_Toc190980675)

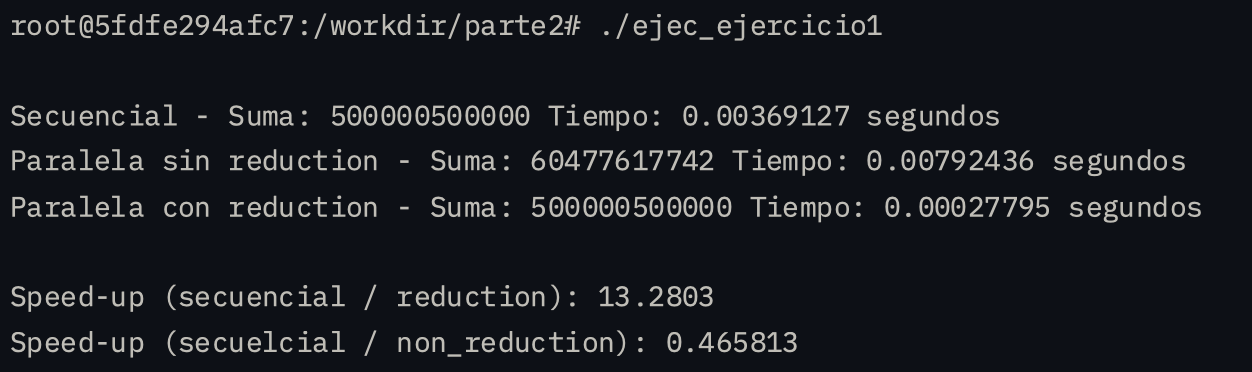
[2. PROGRAMACIÓN CON OPENMP 3](#_Toc190980676)

[CÓDIGO COMENTADO EJERCICIO FINAL 6](#_Toc190980677)

## Desarrollo práctico guiado

## Ejercicio 1:

El objetivo de este ejercicio será comprender como podemos aplicar la cláusula ‘reduction’, que función desempeña en nuestro código y que ventajas obtenemos.

Nuestro código ejecuta 3 bucles con 1000000 iteraciones, 1 de los bucles se ejecuta en modo secuencial, otro de los bucles se ejecuta en paralelo sin hacer uso de la cláusula ‘reduction’ y el ultimo bucle se ejecuta en paralelo también, pero haciendo uso de esta cláusula. Dentro de cada bucle se hace una suma de +i por cada iteración, el tiempo que tarda en ejecutarse cada tipo de bucle se guarda también. Al final calculamos el speed-up entre el secuencial y los bucles paralelos. He obtenido estos resultados:  


Como podemos observar la versión paralela sin reduction nos devuelve una suma invalida, esto se debe a las condiciones de carrera. Las condiciones de carrera ocurren cuando varios hilos intentan modificar una variable compartida de manera desincronizada.

Para paliar esta situación usamos la cláusula ‘reduction’. En nuestro código, en la versión paralela sin reduction, la variable 'suma' es compartida por todos los hilos sin sincronización esto provoca condiciones de carrera y un resultado final erróneo. Por otra parte, con reduction cada hilo mantiene una copia local de la variable 'suma' y al finalizar se combinan, garantizando un resultado correcto sin la sobreescritura simultánea.

En nuestra ejecución hemos obtenido que el bucle paralelo con reduction se ejecuta 13.28 veces más rápido que el bucle secuencial, esto supone una gran mejora de rendimiento. Al tener 16 hilos esta mejora es bastante considerable.

## Ejercicio 2:

El objetivo de este ejercicio es comprender el papel de las técnicas de balanceo de carga y las ventajas que ofrecen unas sobre otras. Cuando se asignan tareas de diferente complejidad a distintos hilos, es común que algunos terminen más rápido y queden inactivos mientras esperan a que otros, con tareas más pesadas, finalicen. Este desequilibrio se puede mitigar utilizando la función omp\_set\_schedule() de OpenMP.

La función omp\_set\_schedule() permite configurar dos parámetros esenciales para la asignación de iteraciones entre hilos:

1. El tipo de tecnica de asignación (schedule), que puede ser, estática, dinámica o guiada.
2. El tamaño del chunk, que define cuántas iteraciones se asignan a un hilo en cada asignación.

**omp\_sched\_static**: Divide las iteraciones de manera fija y equitativa entre los hilos. Es ideal cuando cada iteración tiene un costo de procesamiento similar, ya que minimiza la sobrecarga.  
**omp\_sched\_dynamic**: Asigna bloques de iteraciones de forma dinámica. Cada vez que un hilo termina su bloque, solicita el siguiente, lo que favorece a las aplicaciones con cargas de trabajo variables, aunque introduce una mayor sobrecarga de gestión.  
**omp\_sched\_guided**: Inicialmente asigna bloques grandes y, conforme el trabajo avanza, reduce progresivamente el tamaño de estos bloques. Esto combina menor sobrecarga al inicio con flexibilidad para equilibrar la carga al final de la ejecución.

El programa que hemos compilado y ejecutado para poner en practica estas tecnicas usaba un bucle que se ejecutaba 1 000 000 en el cual se sumaba el cuadrado de la iteración a la suma global de esta forma ‘suma = suma + i \* i’, este bucle se ejecuta 3 veces con las 3 distintas tecnicas de balanceo de carga y se cronometran. Hemos obtenido estos resultados finales:

A screenshot of a computer screen

AI-generated content may be incorrect.

Bajo estos resultados podemos concluir que en este caso las tecnicas dynamic y guided son superiores ya que, como la complejidad del problema se va incrementando en el orden i2, las cargas que soportan los hilos son pesadas y muy variables lo que favorece a estas tecnicas. El tamaño del chunk que hemos usado esta bien, pero si lo hubieramos bajado a 100 se notaria más la diferencia de tiempo entre la static con las tecnicas dynamic y guided al usar menos iteraciones por hilo.