

Tema 8: Introducción a la Programación OpenMP

2020/21

07 de enero de 2021

José Amusquívar Poppe | 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 4	8
Actividad práctica 5	11
Actividad práctica 6	13
Referencias	14

Actividad práctica 1

Se implementa el siguiente código suministrado:

```
#include <omp.h>
#include <stdio.h>

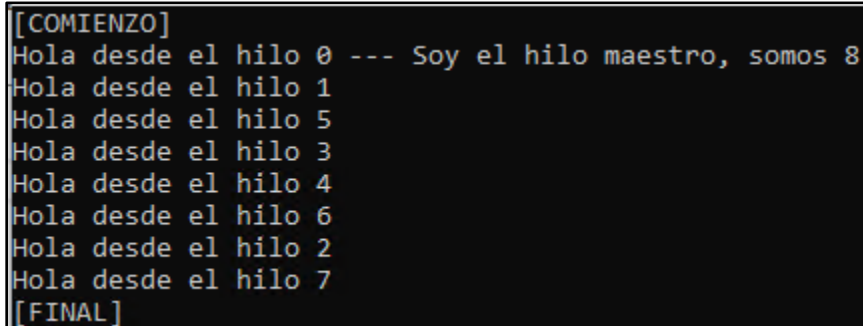
void main()
{
    printf("[COMIENZO]\n");

    #pragma omp parallel num_threads(4)
    {
        int id = omp_get_thread_num();
        int num = omp_get_num_threads();
        printf("Hola desde el hilo %d, somos %d\n", id, num);
    }

    printf("[FINAL]\n");

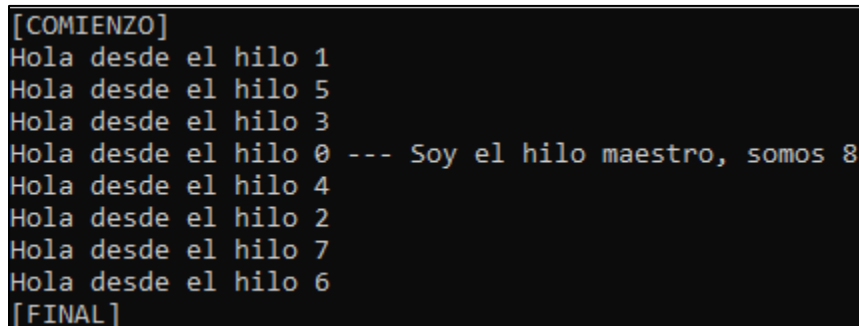
    std::getchar();
    return;
}
```

Se ejecuta el programa varias veces para comprobar que no es determinista, ya que el orden en el que aparecen los mensajes varía en cada ejecución. Se ha comprobado este comportamiento empleando 8 hilos. Además, se ha modificado el programa para que sólo el hilo principal (el hilo 0), sea el que muestre el total de hilos.



```
[COMIENZO]
Hola desde el hilo 0 --- Soy el hilo maestro, somos 8
Hola desde el hilo 1
Hola desde el hilo 5
Hola desde el hilo 3
Hola desde el hilo 4
Hola desde el hilo 6
Hola desde el hilo 2
Hola desde el hilo 7
[FINAL]
```

Figura 1. Primera iteración con 8 hilos. El hilo maestro muestra el total.



```
[COMIENZO]
Hola desde el hilo 1
Hola desde el hilo 5
Hola desde el hilo 3
Hola desde el hilo 0 --- Soy el hilo maestro, somos 8
Hola desde el hilo 4
Hola desde el hilo 2
Hola desde el hilo 7
Hola desde el hilo 6
[FINAL]
```

Figura 2. Segunda iteración, comprobando el cambio de orden en las ejecuciones.

```
[COMIENZO]
Hola desde el hilo 0 --- Soy el hilo maestro, somos 8
Hola desde el hilo 1
Hola desde el hilo 4
Hola desde el hilo 3
Hola desde el hilo 7
Hola desde el hilo 2
Hola desde el hilo 5
Hola desde el hilo 6
[FINAL]
```

Figura 3. Tercera ejecución, comprobando una vez más su aleatoriedad.

Actividad práctica 2

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

Primero se ha comprobado con 4 hilos, obteniendo unos tiempos muy similares, tal como se puede apreciar en la siguiente figura:

```
[COMIENZO]
[Proceso 0] 0.04034410 s!, el resultado es inf
[Proceso 3] 0.04056980 s!, el resultado es inf
[Proceso 1] 0.04182380 s!, el resultado es inf
[Proceso 2] 0.04180060 s!, el resultado es inf
[FINAL]
```

Figura 4. Ejecución del programa con 4 hilos, obteniendo unos tiempos similares.

Posteriormente, se presenta la misma ejecución, pero empleando 8 hilos. Los tiempos obtenidos, una vez más, son similares entre sí, tal como se puede apreciar en la siguiente figura:

```
[COMIENZO]
[Proceso 3] 0.04337610 s!, el resultado es inf
[Proceso 1] 0.04406830 s!, el resultado es inf
[Proceso 2] 0.04364480 s!, el resultado es inf
[Proceso 6] 0.04440960 s!, el resultado es inf
[Proceso 7] 0.04375680 s!, el resultado es inf
[Proceso 5] 0.04511830 s!, el resultado es inf
[Proceso 0] 0.04593160 s!, el resultado es inf
[Proceso 4] 0.04588290 s!, el resultado es inf
[FINAL]
```

Figura 5. Ejecución del programa con 8 hilos, obteniendo unos tiempos similares.

Y, finalmente, se ejecutó el mismo programa, pero con un número de hilos mayor que 8, en este caso, con 11 hilos. Esta vez los tiempos tienen un comportamiento distinto a las anteriores pruebas, tal como se puede apreciar en la siguiente figura:

```
[COMIENZO]
[Proceso 0] 0.04333780 s!, el resultado es inf
[Proceso 4] 0.04434940 s!, el resultado es inf
[Proceso 3] 0.04720190 s!, el resultado es inf
[Proceso 9] 0.04611560 s!, el resultado es inf
[Proceso 5] 0.04841340 s!, el resultado es inf
[Proceso 8] 0.04360130 s!, el resultado es inf
[Proceso 6] 0.04391040 s!, el resultado es inf
[Proceso 1] 0.04893170 s!, el resultado es inf
[Proceso 2] 0.08388170 s!, el resultado es inf
[Proceso 7] 0.08531580 s!, el resultado es inf
[Proceso 10] 0.08686220 s!, el resultado es inf
[FINAL]
```

Figura 6. Ejecución del programa con 11 hilos, comprobando ciertas variaciones.

De la figura anterior se puede deducir que el procesador, únicamente, es capaz de paralelizar 8 procesos. Esto se puede comprobar analizando la figura 6, pues, de los 11 procesos realizados, 8 de ellos tienen un tiempo similar, mientras que los otros 3 restantes poseen el doble de tiempo. Es decir, 8 procesos se han realizado paralelamente, mientras que los 3 restantes se han realizado después de que alguno de los hilos del procesador haya terminado su respectiva tarea.

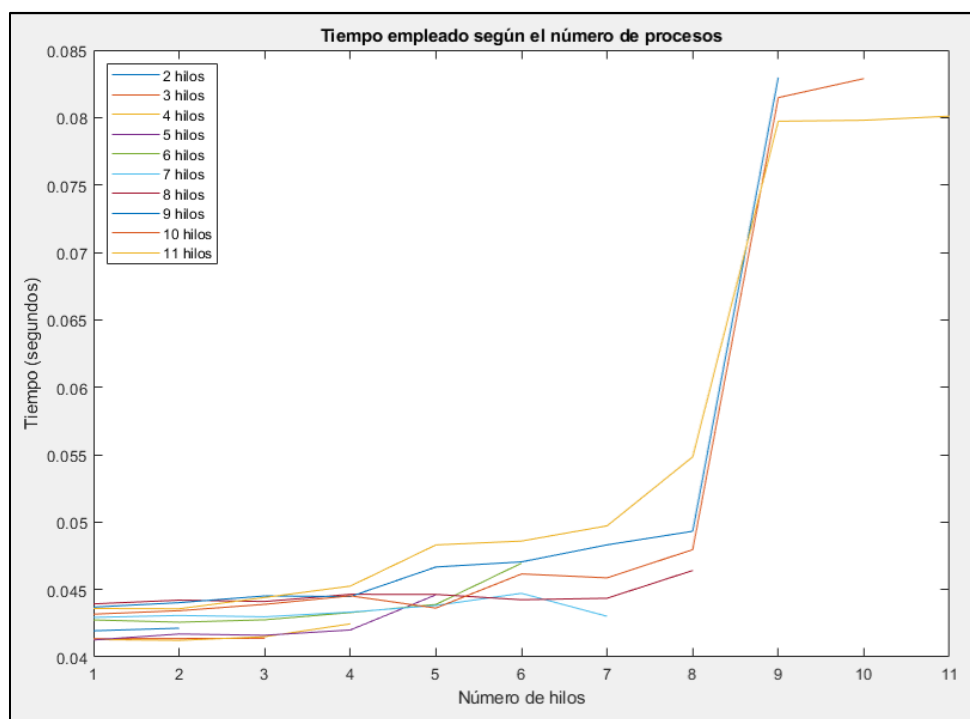


Figura 7. Gráfica que comprueba que, a partir de 8 hilos, el tiempo de ejecución aumenta.

Actividad práctica 3

Se escribe un programa que sume dos vectores de números en coma flotante:

- Se declara tres vectores de 100 elementos
- Se inicializa cada elemento del primero con el valor de su índice
- Se inicializa cada elemento del segundo con el doble del valor de su índice
- Finalmente, se suman los dos vectores en el tercero y comprueba el resultado

```
Indice=17 -> Vect1: 17.000000 Vect2: 34.000000 Vect3: 51.000000
Indice=51 -> Vect1: 51.000000 Vect2: 102.000000 Vect3: 153.000000
Indice=75 -> Vect1: 75.000000 Vect2: 150.000000 Vect3: 225.000000
```

Figura 8. Comprobación de resultados. cada vector obtiene lo que debería.

Se paraleliza el código de forma que haya cuatro hilos entre los que se reparten grupos de 10 iteraciones planificadas de forma dinámica.

```
[Proceso 1] He calculado 33.000000 (indice=11)
[Proceso 2] He calculado 60.000000 (indice=20)
[Proceso 2] He calculado 63.000000 (indice=21)
[Proceso 2] He calculado 66.000000 (indice=22)
[Proceso 2] He calculado 69.000000 (indice=23)
[Proceso 2] He calculado 72.000000 (indice=24)
[Proceso 2] He calculado 75.000000 (indice=25)
[Proceso 2] He calculado 78.000000 (indice=26)
[Proceso 2] He calculado 81.000000 (indice=27)
[Proceso 2] He calculado 84.000000 (indice=28)
[Proceso 2] He calculado 87.000000 (indice=29)
[Proceso 2] He calculado 180.000000 (indice=60)
```

Figura 9. Se comprueba que cada hilo ejecuta 10 iteraciones a la vez.

Se comprueba que cada proceso, en este caso se trata de 4 procesos, ejecuta 10 iteraciones, sin embargo, dado que se ha establecido que tenga un comportamiento dinámico, los hilos no esperarán a que termine uno para poder comenzar su tarea.

```
[Proceso 0] He calculado 0.000000 (indice=0)
[Proceso 2] He calculado 60.000000 (indice=20)
[Proceso 2] He calculado 63.000000 (indice=21)
[Proceso 2] He calculado 66.000000 (indice=22)
[Proceso 2] He calculado 69.000000 (indice=23)
[Proceso 2] He calculado 72.000000 (indice=24)
[Proceso 2] He calculado 75.000000 (indice=25)
[Proceso 0] He calculado 3.000000 (indice=1)
```

Figura 10. Se comprueba que los hilos no esperan a que otro termine.

Además, se modifica el programa para que cada hilo muestre qué elementos del vector resultado ha calculado, comprobando las variaciones que se producen en diferentes ejecuciones. Esto puede ser comprobado mediante el índice del vector resultado que ha operado el hilo.

```
[Proceso 3] He calculado 270.000000 (indice=90)
[Proceso 3] He calculado 273.000000 (indice=91)
[Proceso 3] He calculado 276.000000 (indice=92)
[Proceso 3] He calculado 279.000000 (indice=93)
[Proceso 3] He calculado 282.000000 (indice=94)
[Proceso 3] He calculado 285.000000 (indice=95)
[Proceso 3] He calculado 288.000000 (indice=96)
[Proceso 3] He calculado 291.000000 (indice=97)
[Proceso 3] He calculado 294.000000 (indice=98)
[Proceso 3] He calculado 297.000000 (indice=99)
[Proceso 2] He calculado 105.000000 (indice=35)
[Proceso 2] He calculado 108.000000 (indice=36)
[Proceso 2] He calculado 111.000000 (indice=37)
[Proceso 2] He calculado 114.000000 (indice=38)
[Proceso 2] He calculado 117.000000 (indice=39)
```

Figura 11. Se comprueban los índices que opera cada hilo.

```
[Proceso 0] He calculado 270.000000 (indice=90)
[Proceso 0] He calculado 273.000000 (indice=91)
[Proceso 0] He calculado 276.000000 (indice=92)
[Proceso 0] He calculado 279.000000 (indice=93)
[Proceso 0] He calculado 282.000000 (indice=94)
[Proceso 0] He calculado 285.000000 (indice=95)
[Proceso 0] He calculado 288.000000 (indice=96)
[Proceso 0] He calculado 291.000000 (indice=97)
[Proceso 0] He calculado 294.000000 (indice=98)
[Proceso 0] He calculado 297.000000 (indice=99)
[Proceso 3] He calculado 90.000000 (indice=30)
[Proceso 3] He calculado 93.000000 (indice=31)
[Proceso 3] He calculado 96.000000 (indice=32)
[Proceso 3] He calculado 99.000000 (indice=33)
[Proceso 3] He calculado 102.000000 (indice=34)
[Proceso 3] He calculado 105.000000 (indice=35)
[Proceso 3] He calculado 108.000000 (indice=36)
[Proceso 3] He calculado 111.000000 (indice=37)
[Proceso 3] He calculado 114.000000 (indice=38)
[Proceso 3] He calculado 117.000000 (indice=39)
```

Figura 12. Se comprueba en otra ejecución que los índices e hilos varían.

Con lo que se puede comprobar que cada hilo ha operado distintos elementos del vector en distintas ejecuciones del programa.

Actividad práctica 4

Se modifica el programa anterior para que la planificación se realice de forma estática y ejecutándolo varias veces para comprobar los cambios que se producen en su comportamiento.

```
[Proceso 3] He calculado 237.000000 (indice=79)
[Proceso 2] He calculado 60.000000 (indice=20)
[Proceso 2] He calculado 63.000000 (indice=21)
[Proceso 2] He calculado 66.000000 (indice=22)
[Proceso 2] He calculado 69.000000 (indice=23)
[Proceso 2] He calculado 72.000000 (indice=24)
[Proceso 2] He calculado 75.000000 (indice=25)
[Proceso 2] He calculado 78.000000 (indice=26)
[Proceso 2] He calculado 81.000000 (indice=27)
[Proceso 2] He calculado 84.000000 (indice=28)
[Proceso 2] He calculado 87.000000 (indice=29)
[Proceso 2] He calculado 180.000000 (indice=60)
```

Figura 13. Se comprueba que cada hilo realiza 10 iteraciones.

Cambiando el comportamiento de los hilos de dinámico (“dynamic”) a estático (“static”), se consigue que cada hilo realice sus 10 iteraciones sin interrupciones, por lo que si hay hilos que deseen realizar sus operaciones, éstos deben esperar a que el hilo actual termine las suyas.

Se añade al programa las instrucciones necesarias para medir el tiempo de ejecución de la suma de los dos vectores y comprobando las diferencias que se producen. Se añade al programa las instrucciones necesarias para medir el tiempo de ejecución de la suma de los dos vectores y comprobando las diferencias que se producen.

- Se utiliza varios tamaños de vector (100, 1000, 10000...)
- Se utiliza varios tamaños de grupo de iteraciones (10, 100...)

```
Tiempo que tarda con 10 elementos y grupo de 10 iteraciones: 0.000466
Tiempo que tarda con 100 elementos y grupo de 10 iteraciones: 0.000007
Tiempo que tarda con 1000 elementos y grupo de 10 iteraciones: 0.000007
Tiempo que tarda con 10000 elementos y grupo de 10 iteraciones: 0.000016
Tiempo que tarda con 100000 elementos y grupo de 10 iteraciones: 0.000700
Tiempo que tarda con 1000000 elementos y grupo de 10 iteraciones: 0.004431
Tiempo que tarda con 10000000 elementos y grupo de 10 iteraciones: 0.050151
Tiempo que tarda con 100000000 elementos y grupo de 10 iteraciones: 0.521180
```

Figura 14. Comprobación empleando conjuntos de 10 iteraciones.

```
Tiempo que tarda con 10 elementos y grupo de 1000 iteraciones: 0.000313
Tiempo que tarda con 100 elementos y grupo de 1000 iteraciones: 0.000008
Tiempo que tarda con 1000 elementos y grupo de 1000 iteraciones: 0.000007
Tiempo que tarda con 10000 elementos y grupo de 1000 iteraciones: 0.000016
Tiempo que tarda con 100000 elementos y grupo de 1000 iteraciones: 0.000088
Tiempo que tarda con 1000000 elementos y grupo de 1000 iteraciones: 0.002808
Tiempo que tarda con 10000000 elementos y grupo de 1000 iteraciones: 0.022981
Tiempo que tarda con 100000000 elementos y grupo de 1000 iteraciones: 0.227052
```

Figura 15. Comprobación empleando conjuntos de 1000 iteraciones.

Finalmente, se dibuja una gráfica que detalle las diferencias encontradas. Dado que existen dos variables independientes, el tamaño del vector y del conjunto de iteraciones (“CHUNK”), se ha realizado un total de ocho gráficas. Se ha decidido que el tamaño del vector dentro de cada gráfica, mientras el tamaño del conjunto de iteraciones, únicamente, varía al realizar una nueva gráfica.

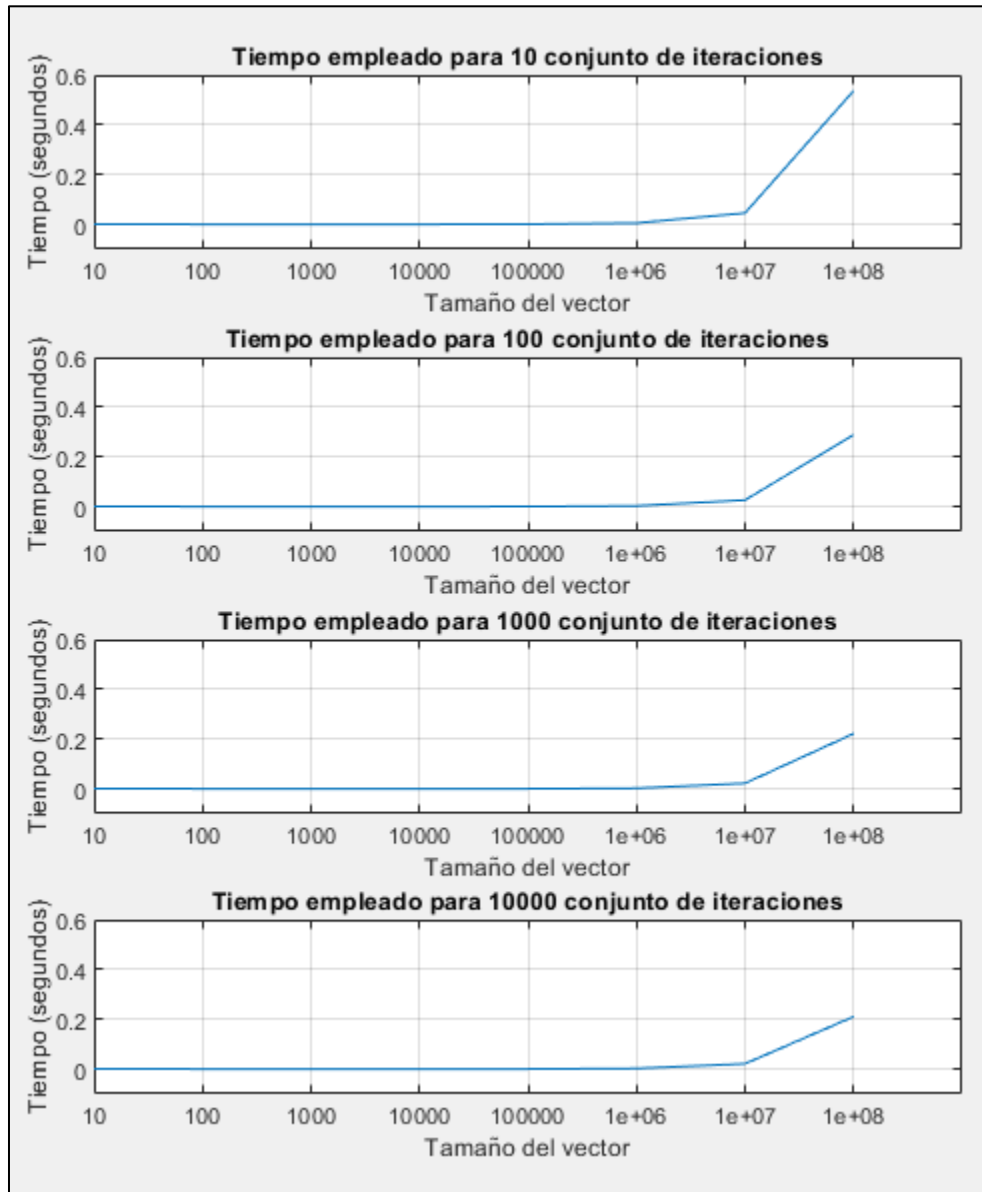


Figura 16. Gráficas variando el tamaño del vector y del conjunto de iteraciones (1).

Como se puede apreciar en la figura anterior, a medida que aumenta el número del conjunto de iteraciones, el tiempo empleado disminuye. Esto es lógico pues, al disponer únicamente de 10 hilos y tener establecido el comportamiento de los hilos en estático (“static”), si se divide todo el vector en trozos muy pequeños, cada hilo tendrá que realizar más tareas, generando así más tiempo de espera entre hilos, ya que cada hilo en cola deberá esperar a que termine otro para poder realizar

sus tareas encomendadas. A medida que se aumente el conjunto de iteraciones, la suma se subdivide en trozos más pequeños a cada hilo por lo que el tiempo disminuye.

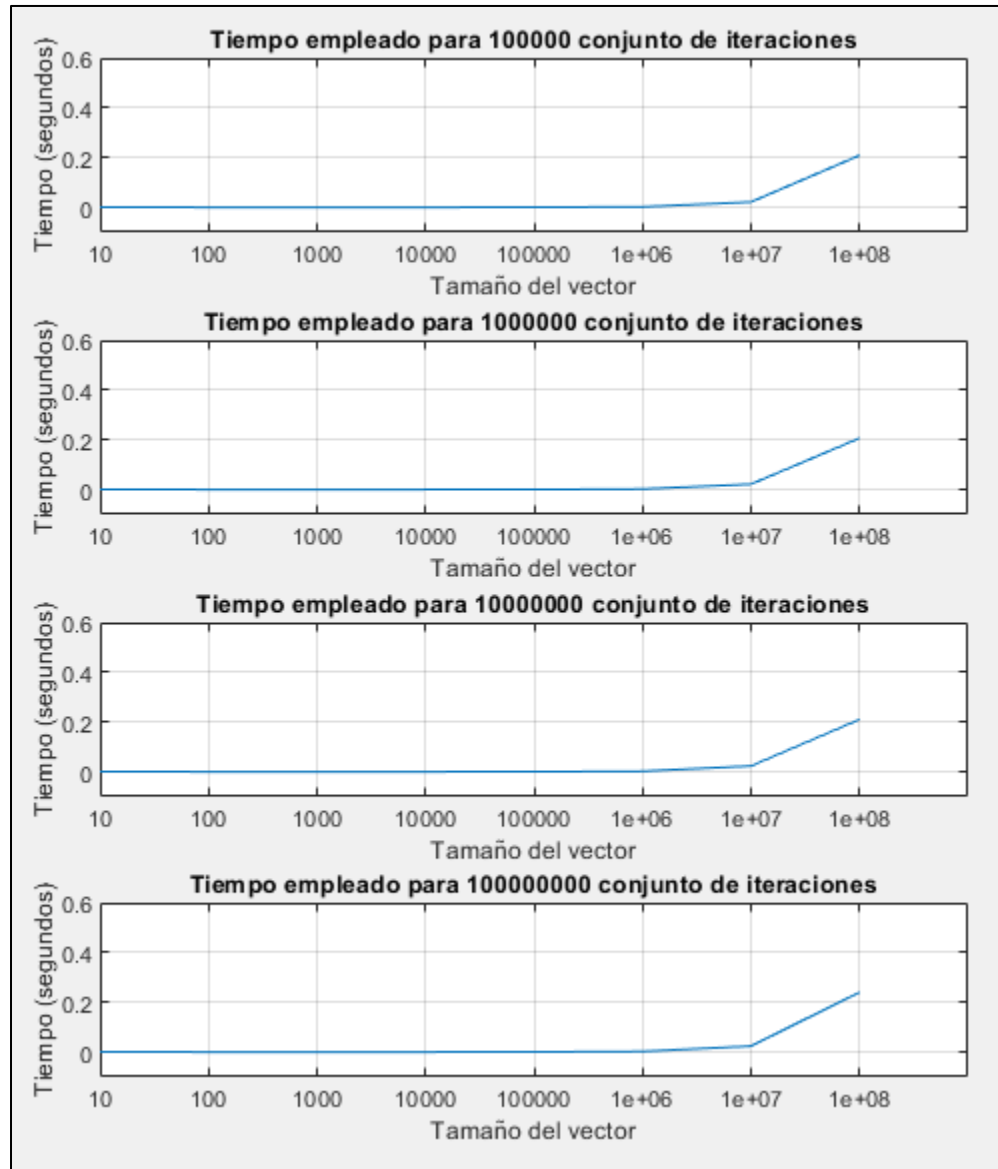


Figura 17. Gráficas variando el tamaño del vector y del conjunto de iteraciones (2).

Esta segunda gráfica es una continuación a la figura 16, en la que se sigue aumentando el número del conjunto de iteraciones llegando, inclusive, a establecer un número de conjunto de iteraciones igual al tamaño del vector. Se observa que a partir de un cierto valor de número del conjunto de iteraciones el tiempo se mantiene en unos valores parecidos y se comprueba en las últimas gráficas el tiempo empleado comienza a incrementar.

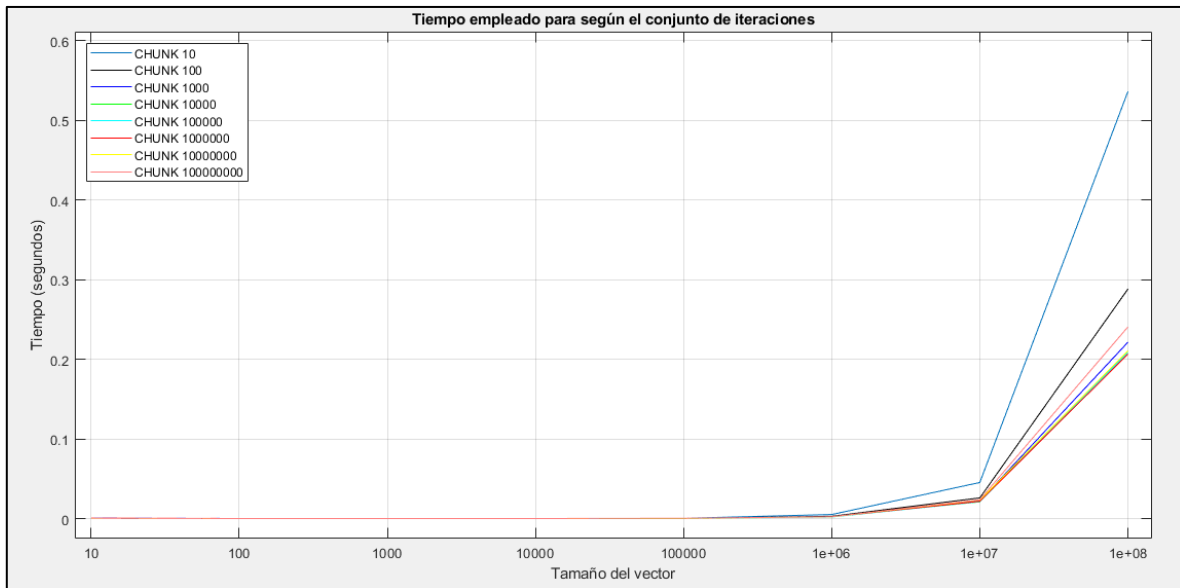


Figura 18. Gráfica general

Finalmente, se compara todas las subgráficas anteriores en una única gráfica general.

Actividad práctica 5

Se escribe un programa que suma y multiplica, de forma separada, los elementos de dos vectores de números en coma flotante:

- Se declara cuatro vectores de 100 elementos
- Se inicializa cada elemento del primero con el valor de su índice
- Se inicializa cada elemento del segundo con el doble del valor de su índice
- Se suma los dos vectores en el tercero y se comprueba el resultado
- Se multiplica los dos vectores en el cuarto y se comprueba el resultado

```
Indice=17 -> Vect1: 17.000000    Vect2: 34.000000
             Suma: 51.000000    Multiplicacion: 578.000000
Indice=51 -> Vect1: 51.000000    Vect2: 102.000000
             Suma: 153.000000    Multiplicacion: 5202.000000
Indice=75 -> Vect1: 75.000000    Vect2: 150.000000
             Suma: 225.000000    Multiplicacion: 11250.000000
```

Figura 19. Se comprueba que los vectores están bien creados.

Se paraleliza el código de forma que haya dos secciones; la primera sección realizará la suma y la segunda sección realizará la multiplicación. Además, se modifica el programa para que cada hilo muestre qué elementos de cada vector resultado ha calculado y se comprueba las variaciones que se producen en diferentes ejecuciones.

Se puede apreciar que en la figura anterior el hilo principal, (el 0), es el encargado de realizar la suma mientras que el hilo 2 es el que realiza la multiplicación.

```
[Proceso 1] (Multiplicacion) He calculado 16928.000000 (indice=92)
[Proceso 1] (Multiplicacion) He calculado 17298.000000 (indice=93)
[Proceso 1] (Multiplicacion) He calculado 17672.000000 (indice=94)
[Proceso 1] (Multiplicacion) He calculado 18050.000000 (indice=95)
[Proceso 1] (Multiplicacion) He calculado 18432.000000 (indice=96)
[Proceso 1] (Multiplicacion) He calculado 18818.000000 (indice=97)
[Proceso 1] (Multiplicacion) He calculado 19208.000000 (indice=98)
[Proceso 1] (Multiplicacion) He calculado 19602.000000 (indice=99)
[Proceso 0] (Suma) He calculado 96.000000 (indice=32)
[Proceso 0] (Suma) He calculado 99.000000 (indice=33)
[Proceso 0] (Suma) He calculado 102.000000 (indice=34)
[Proceso 0] (Suma) He calculado 105.000000 (indice=35)
[Proceso 0] (Suma) He calculado 108.000000 (indice=36)
[Proceso 0] (Suma) He calculado 111.000000 (indice=37)
[Proceso 0] (Suma) He calculado 114.000000 (indice=38)
[Proceso 0] (Suma) He calculado 117.000000 (indice=39)
[Proceso 0] (Suma) He calculado 120.000000 (indice=40)
```

Figura 20. Otra comprobación de la suma y la multiplicación.

```
[Proceso 1] (Multiplicacion) He calculado 2.000000 (indice=1)
[Proceso 1] (Multiplicacion) He calculado 8.000000 (indice=2)
[Proceso 1] (Multiplicacion) He calculado 18.000000 (indice=3)
[Proceso 1] (Multiplicacion) He calculado 32.000000 (indice=4)
[Proceso 1] (Multiplicacion) He calculado 50.000000 (indice=5)
[Proceso 1] (Multiplicacion) He calculado 72.000000 (indice=6)
[Proceso 1] (Multiplicacion) He calculado 98.000000 (indice=7)
```

Figura 21. Se comprueba que cada hilo ejecuta o la suma o la multiplicación.

Tal como se aprecia en la figura 19 y 20, el hilo encargado de realizar la suma, en todas las ejecuciones que se han realizado, ha sido siempre el hilo 0, mientras que la el hilo 1 realiza la multiplicación. Además, dado que sólo existen dos secciones en el programa, el número de hilos máximo que se usarán será también de dos.

Actividad práctica 6

¿Cómo se comporta el programa anterior si hay menos hilos que secciones?

Puesto que cada sección es ejecutada, únicamente, por un hilo, si existen menos hilos que secciones, existirán hilos que realicen más de una sección. Por ejemplo, en el caso anterior, en el que el número de secciones es igual a 2, si sólo existiese un hilo (hilo 0), éste sería el encargado de realizar tanto la suma como la multiplicación.

```
[Proceso 0] (Suma) He calculado 285.000000 (indice=95)
[Proceso 0] (Suma) He calculado 288.000000 (indice=96)
[Proceso 0] (Suma) He calculado 291.000000 (indice=97)
[Proceso 0] (Suma) He calculado 294.000000 (indice=98)
[Proceso 0] (Suma) He calculado 297.000000 (indice=99)
[Proceso 0] (Multiplicacion) He calculado 0.000000 (indice=0)
[Proceso 0] (Multiplicacion) He calculado 2.000000 (indice=1)
[Proceso 0] (Multiplicacion) He calculado 8.000000 (indice=2)
[Proceso 0] (Multiplicacion) He calculado 18.000000 (indice=3)
[Proceso 0] (Multiplicacion) He calculado 32.000000 (indice=4)
```

Figura 22. Se comprueba que el hilo principal realiza ambas operaciones.

¿Cómo se comporta el programa anterior si hay más hilos que secciones?

Tal como se comentó en la pregunta anterior, debido a que cada sección es ejecutada, únicamente, por un hilo, si se indica un número mayor de hilos que supere a las secciones, el programa ignora los hilos sobrantes. Por tanto, el número de hilos debería ser menor o igual al número de secciones.

Finalmente, se modifica el programa para que haga un uso adecuado de cuatro hilos. Para ello se ha dividido ambos vectores para tener 4 trozos con el fin de crear 4 secciones que realicen tanto la suma como la multiplicación. Vemos a continuación que cada sección es ejecutada por un hilo distinto:

```
[Proceso 0] (Suma 1) He calculado 141.000000 (indice=47)
[Proceso 0] (Suma 1) He calculado 144.000000 (indice=48)
[Proceso 0] (Suma 1) He calculado 147.000000 (indice=49)
[Proceso 3] (Multiplicacion 2) He calculado 5000.000000 (indice=50)
[Proceso 3] (Multiplicacion 2) He calculado 5202.000000 (indice=51)
[Proceso 3] (Multiplicacion 2) He calculado 5408.000000 (indice=52)
```

Figura 23. Primera parte de la suma y segunda parte de la multiplicación.

```
[Proceso 3] (Multiplicacion 2) He calculado 18818.000000 (indice=97)
[Proceso 3] (Multiplicacion 2) He calculado 19208.000000 (indice=98)
[Proceso 3] (Multiplicacion 2) He calculado 19602.000000 (indice=99)
[Proceso 2] (Suma 2) He calculado 150.000000 (indice=50)
[Proceso 2] (Suma 2) He calculado 153.000000 (indice=51)
[Proceso 2] (Suma 2) He calculado 156.000000 (indice=52)
```

Figura 24. Segunda parte de la multiplicación y segunda parte de la suma.

```
[Proceso 2] (Suma 2) He calculado 291.000000 (indice=97)
[Proceso 2] (Suma 2) He calculado 294.000000 (indice=98)
[Proceso 2] (Suma 2) He calculado 297.000000 (indice=99)
[Proceso 1] (Multiplicacion 1) He calculado 0.000000 (indice=0)
[Proceso 1] (Multiplicacion 1) He calculado 2.000000 (indice=1)
[Proceso 1] (Multiplicacion 1) He calculado 8.000000 (indice=2)
```

Figura 25. Segunda parte de la suma y primera parte de la multiplicación.

Referencias

ULPGC. (s.f.). Obtenido de https://ncvt-aep.ulpgc.es/cv/ulpgctp21/pluginfile.php/412007/mod_resource/content/0/8.%20Introducci%C3%B3n%20a%20la%20Programaci%C3%B3n%20OpenMP.pdf