Arquitectura de los computadores

Práctica de programación paralela con OpenMP

22 de noviembre de 2019

Autores:
Juan Francisco García Casado
100383464@alumnos.uc3m.es

Guillermo Bautista-Abad Acebo
100383477@alumnos.uc3m.es

Alejandro de la Cruz Alvarado
100383497@alumnos.uc3m.es

Índice

1. Secuencial **3**
1.1. Descripción del programa . . . 3
1.2. Pruebas de rendimiento . . . 3
1.2.1. Prámetros de la máquina empleada . . . 3
1.2.2. Pruebas . . . 4
1.2.3. Análisis de resultados . . . 6

2. Versión Paralelo **6**
2.1. Introducción . . . 6
2.2. Pruebas de rendimiento . . . 7
2.2.1. Prámetros de la máquina empleada . . . 7
2.3. Pruebas . . . 8
2.3.1. 1 hilo . . . 8
2.3.2. 2 hilos . . . 10
2.3.3. 4 hilos . . . 12
2.3.4. 8 hilos . . . 14
2.3.5. 16 hilos . . . 16
2.4. Análisis de resultados . . . 18
2.5. Impacto de la planificación . . . 18
2.5.1. 4 Hilos . . . 18
2.5.2. 8 Hilos . . . 19

3. Plan de pruebas **19**
3.1. Versión secuencial . . . 19
3.2. Versión paralela . . . 20

4. Conclusión **20**

1. Secuencial

1.1. Descripción del programa

En este apartado de la práctica se nos pide que realicemos la simulación del movimiento, atracción y rebote de los asteroides de forma secuencial. Esto quiere decir que cada cálculo debe realizarse uno detrás de otro.

Para representar los asteroides y los planetas, hemos utilizado `structs` que contengan sus datos como la posición, y la velocidad y el vector de fuerzas en caso de los asteroides. La forma en la que hemos calculado la atracción entre cuerpos es mediante el uso de bucles *for* anidados que recorren un `array` de asteroides y planetas, y añaden el resultado al vector de fuerzas. Una vez se han calculado todas las fuerzas, para cada asteroide hemos obtenido el sumatorio y con este hemos calculado la velocidad, aceleración y posición. Finalmente hemos comprobado los rebotes con los bordes y entre asteroides.

Adicionalmente, tal y como pide el enunciado, al iniciar la ejecución se escriben los parámetros de inicio en un archivo *init_conf.txt* y una vez termina la ejecución escribe los parámetros finales de exclusivamente los asteroides (ya que los planetas no han variado) en un archivo *out.txt*

1.2. Pruebas de rendimiento

1.2.1. Prámetros de la máquina empleada

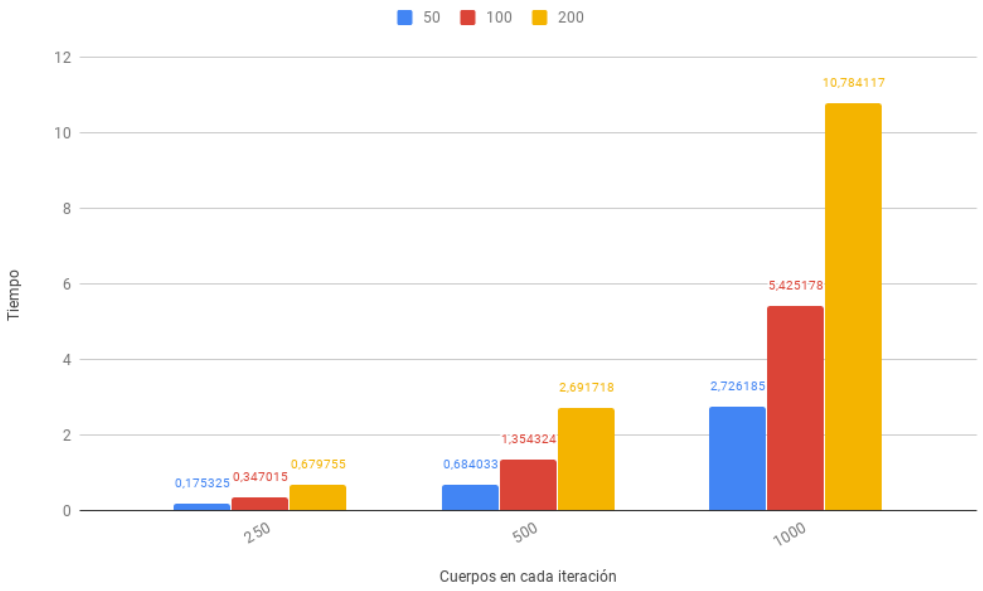
1. Procesador → AMD Ryzen 5 1600 Six-Core Processor
2. Núcleos → 12
3. Hilos → 24
4. Memoria Principal → 16032 MiB
5. Memoria Caché:
   a) L1d → 192 KiB
   b) L1i → 384 KiB
   c) L2 → 3 MiB

d) L3 → 16 MiB

6. Sistema Operativo → ArchLabs Linux x86_64
7. Versión gcc → 9.1.0

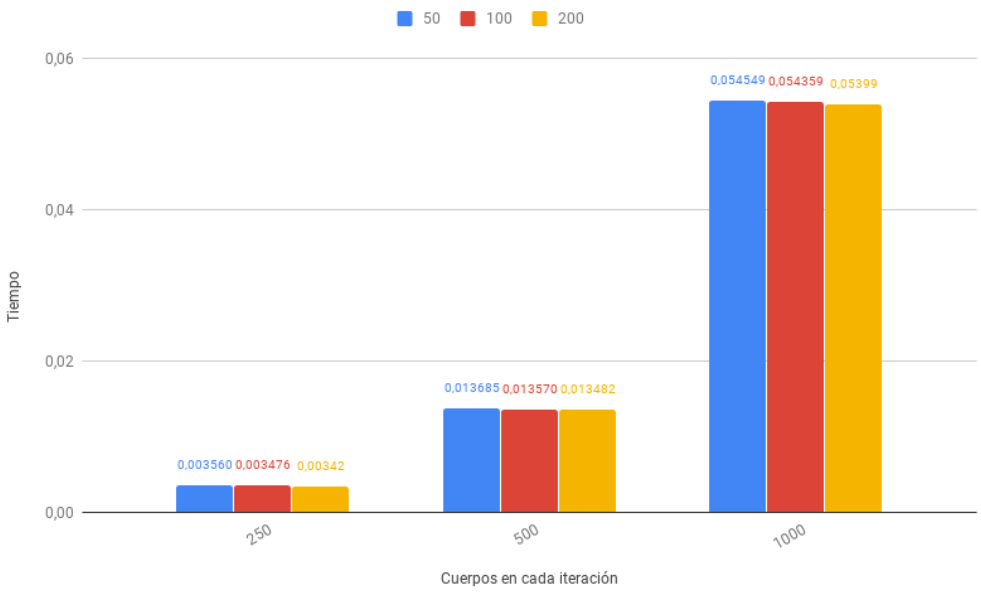
1.2.2. Pruebas

Tiempo medio por ejecución



Asteroides Iteraciones Planetas → Tiempo
250 50 250 → 0,175325 s
250 100 250 → 0,347015 s
250 200 250 → 0,679755 s
500 50 500 → 0,684033 s
500 100 500 → 1,354324 s
500 200 500 → 2,691718 s
1000 50 1000 → 2,726185 s
1000 100 1000 → 5,425178 s
1000 200 1000 → 10,784117 s

Tiempo medio por iteración



Asteroides Iteraciones Planetas → Tiempo
250 50 250 → 0,003560 s
250 100 250 → 0,003476 s
250 200 250 → 0,003420 s
500 50 500 → 0,013685 s
500 100 500 → 0,013570 s
500 200 500 → 0,013482 s
1000 50 1000 → 0,054549 s
1000 100 1000 → 0,054359 s
1000 200 1000 → 0,053990 s

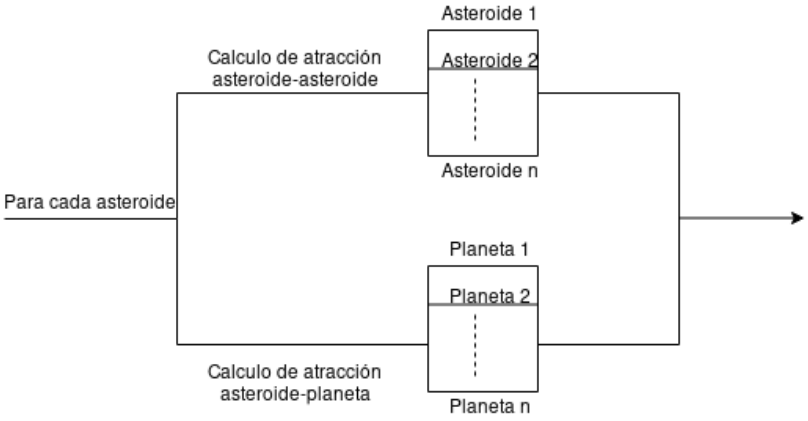
1.2.3. Análisis de resultados

Conforme aumenta el número de asteroides, planetas, o iteraciones el número de cálculos es mayor y por lo tanto el tiempo de ejecución es mayor. Lo mas remarcable de estas pruebas es que el tiempo de ejecución por cada iteración no varía, a diferencia de lo que veremos mas adelante en las pruebas paralelas. Estas gráficas tienen sentido como comparación de las paralelas.

2. Versión Paralelo

2.1. Introducción

Para realizar esta parte hemos cogido el código secuencial y con la librería OpenMP hemos añadido paralelismo, primero a la hora de calcular la atracción entre cuerpos, implementando paralelismo en el bucle for que itera sobre cada asteroide, y dentro de cada iteración de este mismo bucle siguiendo el siguiente esquema de paralelismo:



Después hemos paralelizado el sumatorio de fuerzas, implementando paralelismo en el bucle que recorre los asteroides, como el que recorre el vector de fuerzas de cada asteroides, utilizando un `atomic` para el sumatorio sobre la variable. Y por último hemos paralelizado el bucle que actualiza la posición de los asteroides, siendo esto posible al haber separado el bucle que hace el sumatorio de fuerzas y calcula la velocidad, del que actualiza la posición.

2.2. Pruebas de rendimiento

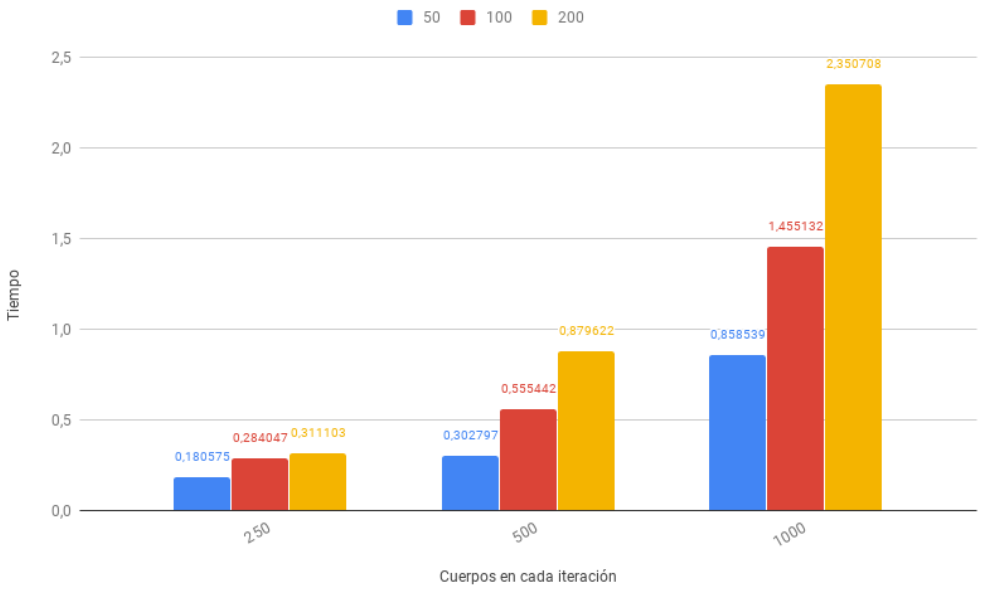
2.2.1. Prámetros de la máquina empleada

1. Procesador → AMD Ryzen 5 1600 Six-Core Processor
2. Núcleos → 12
3. Hilos → 24
4. Memoria Principal → 16032 MiB
5. Memoria Caché:
   a) L1d → 192 KiB
   b) L1i → 384 KiB
   c) L2 → 3 MiB
   d) L3 → 16 MiB
6. Sistema Operativo → ArchLabs Linux x86_64
7. Versión gcc → 9.1.0

2.3. Pruebas

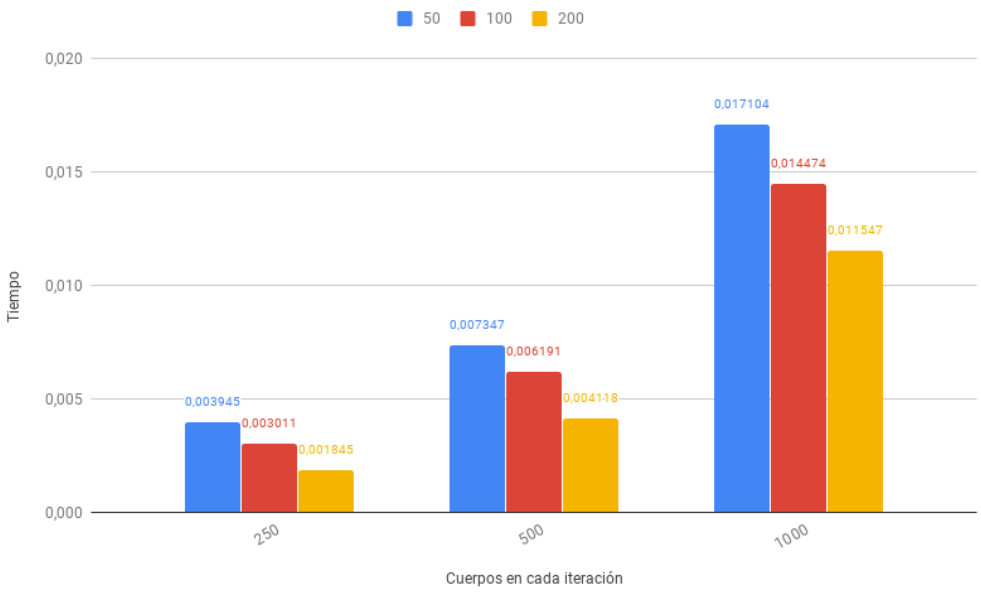
2.3.1. 1 hilo

Tiempo medio por ejecución



Asteroides Iteraciones Planetas → Tiempo
250 50 250 → 0,180575 s
250 100 250 → 0,284047 s
250 200 250 → 0,311103 s
500 50 500 → 0,302797 s
500 100 500 → 0,555442 s
500 200 500 → 0,879622 s
1000 50 1000 → 0,858539 s
1000 100 1000 → 1,455132 s
1000 200 1000 → 2,350708 s

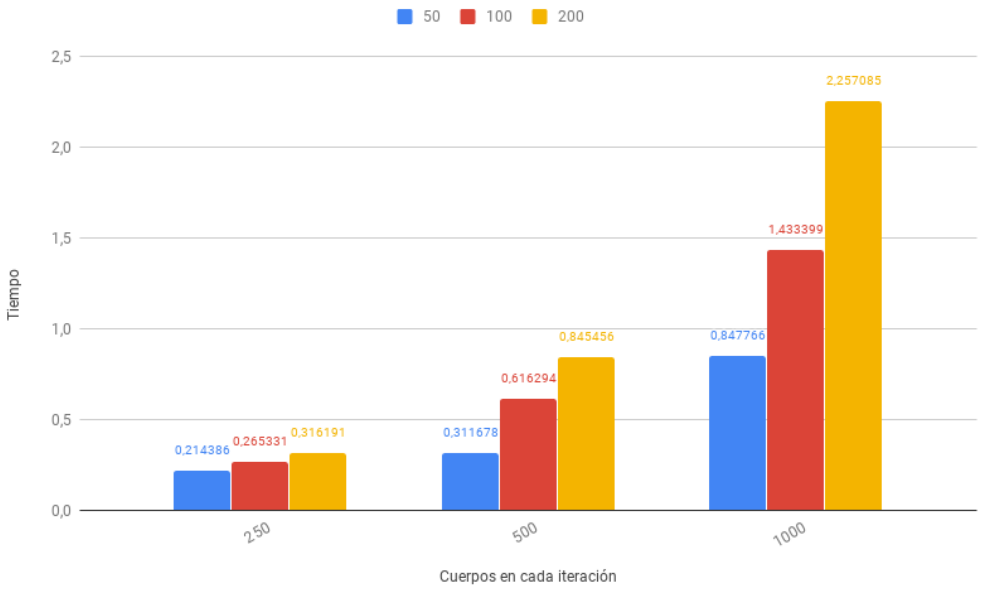
Tiempo medio por iteración



Asteroides Iteraciones Planetas → Tiempo
250 50 250 → 0,003945 s
250 100 250 → 0,003011 s
250 200 250 → 0,001845 s
500 50 500 → 0,007347 s
500 100 500 → 0,006191 s
500 200 500 → 0,004118 s
1000 50 1000 → 0,017104 s
1000 100 1000 → 0,014474 s
1000 200 1000 → 0,011547 s

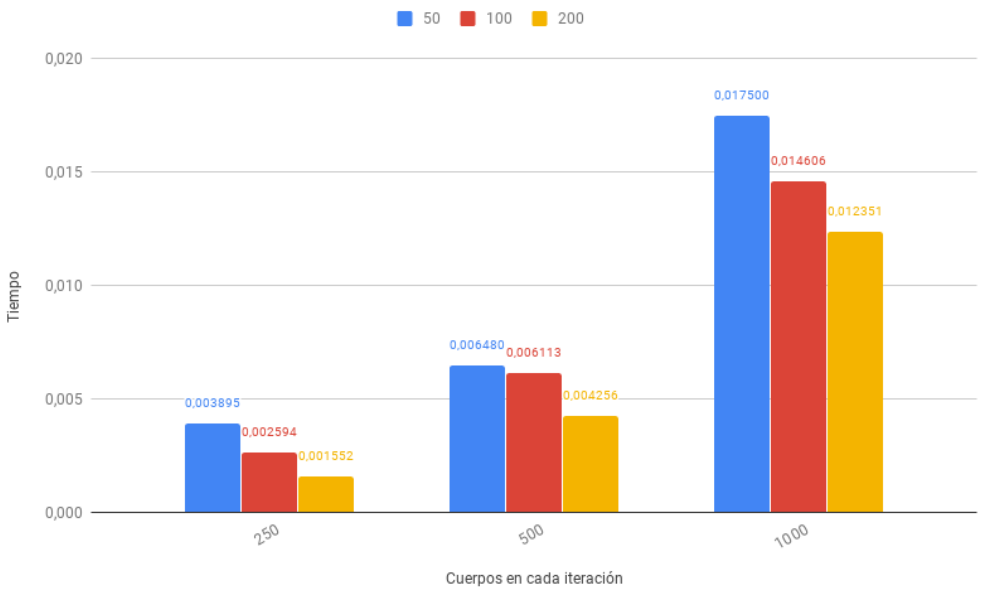
2.3.2. 2 hilos

Tiempo medio por ejecución



Asteroides Iteraciones Planetas → Tiempo
250 50 250 → 0,214386 s
250 100 250 → 0,265331 s
250 200 250 → 0,316191 s
500 50 500 → 0,311678 s
500 100 500 → 0,616294 s
500 200 500 → 0,845456 s
1000 50 1000 → 0,847766 s
1000 100 1000 → 1,433399 s
1000 200 1000 → 2,257085 s

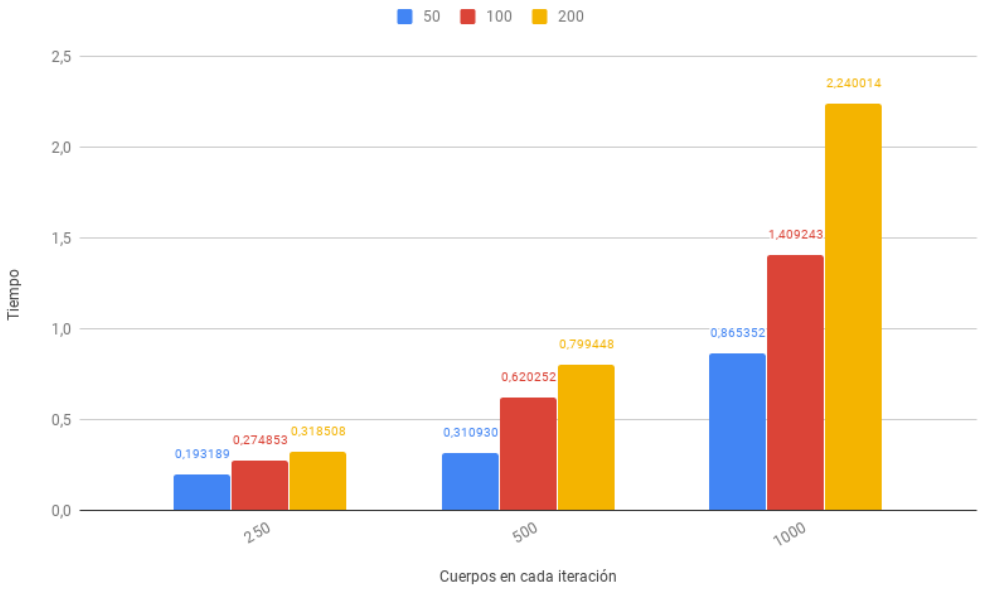
Tiempo medio por iteración



Asteroides Iteraciones Planetas → Tiempo
250 50 250 → 0,003895 s
250 100 250 → 0,002594 s
250 200 250 → 0,001552 s
500 50 500 → 0,006480 s
500 100 500 → 0,006113 s
500 200 500 → 0,004256 s
1000 50 1000 → 0,017500 s
1000 100 1000 → 0,014606 s
1000 200 1000 → 0,012351 s

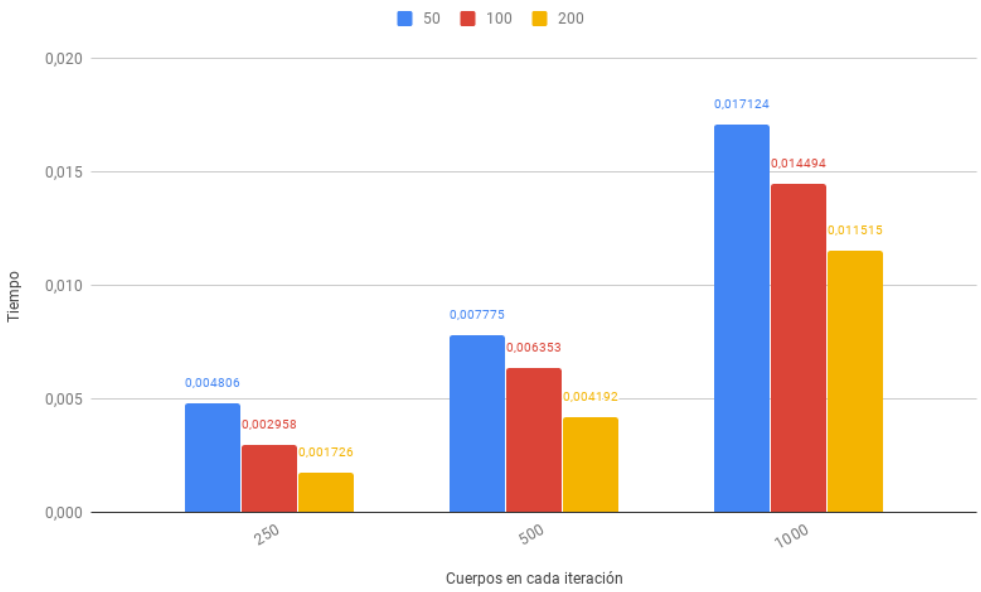
2.3.3. 4 hilos

Tiempo medio por ejecución



Asteroides Iteraciones Planetas → Tiempo
250 50 250 → 0,193189 s
250 100 250 → 0,274853 s
250 200 250 → 0,318508 s
500 50 500 → 0,310930 s
500 100 500 → 0,620252 s
500 200 500 → 0,799448 s
1000 50 1000 → 0,865352 s
1000 100 1000 → 1,409243 s
1000 200 1000 → 2,240014 s

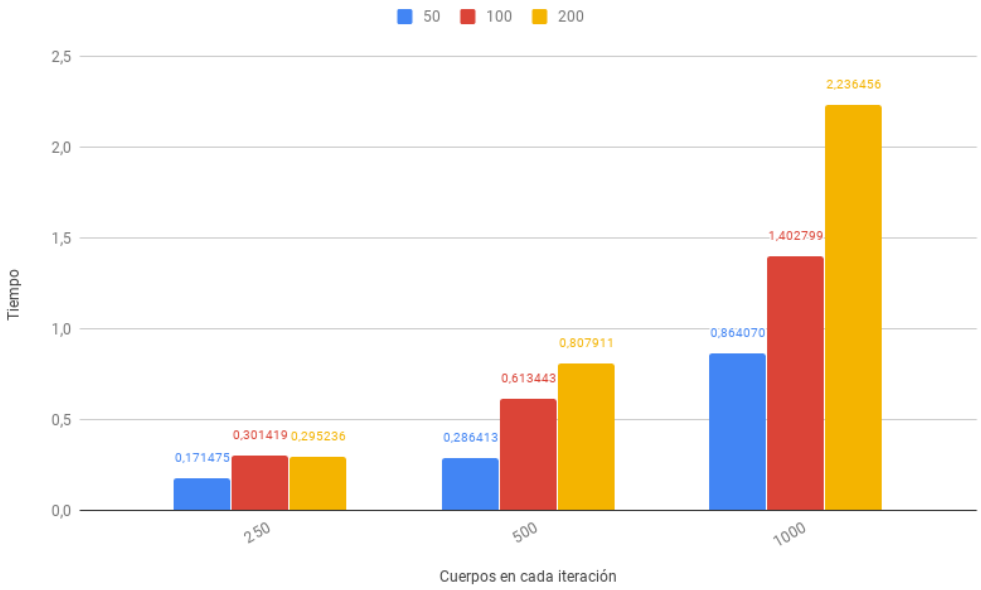
Tiempo medio por iteración



Asteroides Iteraciones Planetas → Tiempo
250 50 250 → 0,004806 s
250 100 250 → 0,002958 s
250 200 250 → 0,001726 s
500 50 500 → 0,007775 s
500 100 500 → 0,006353 s
500 200 500 → 0,004192 s
1000 50 1000 → 0,017124 s
1000 100 1000 → 0,014494 s
1000 200 1000 → 0,011515 s

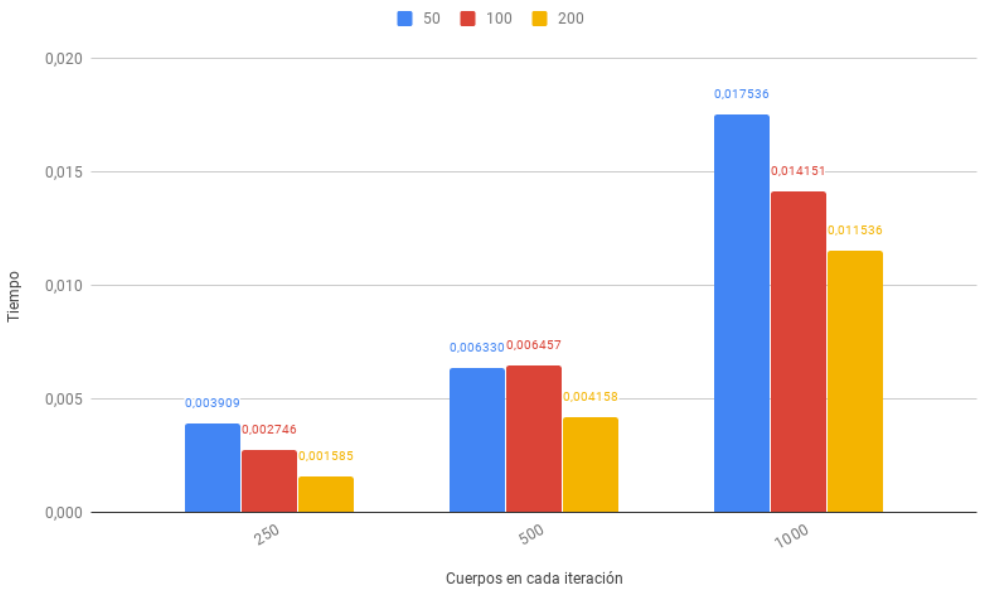
2.3.4. 8 hilos

Tiempo medio por ejecución



Asteroides Iteraciones Planetas → Tiempo
250 50 250 → 0,171475 s
250 100 250 → 0,301419 s
250 200 250 → 0,295236 s
500 50 500 → 0,286413 s
500 100 500 → 0,613443 s
500 200 500 → 0,807911 s
1000 50 1000 → 0,864070 s
1000 100 1000 → 1,402799 s
1000 200 1000 → 2,236456 s

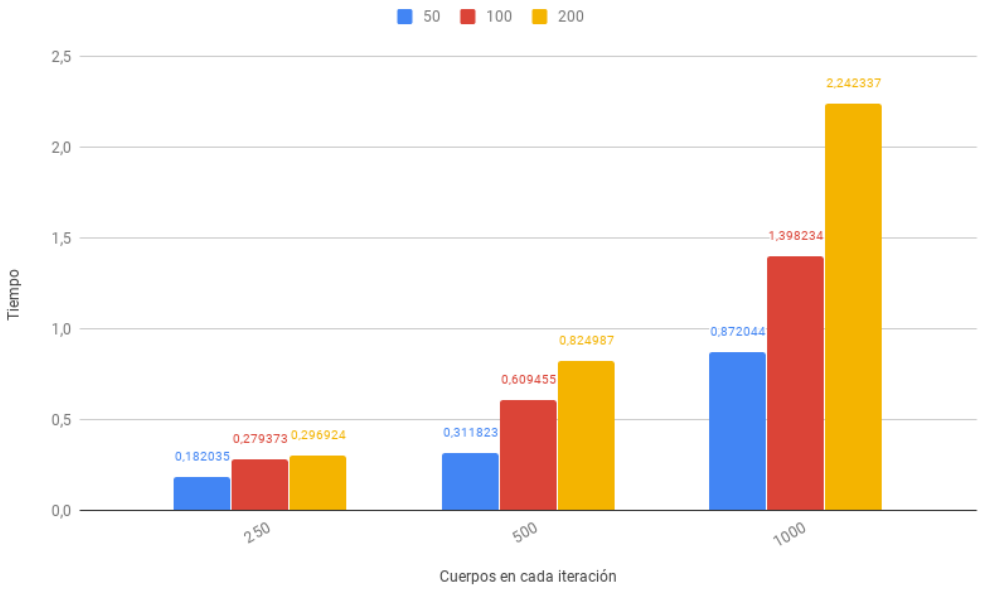
Tiempo medio por iteración



Asteroides Iteraciones Planetas → Tiempo
250 50 250 → 0,003909 s
250 100 250 → 0,002746 s
250 200 250 → 0,001585 s
500 50 500 → 0,006330 s
500 100 500 → 0,006457 s
500 200 500 → 0,004158 s
1000 50 1000 → 0,017536 s
1000 100 1000 → 0,014151 s
1000 200 1000 → 0,011536 s

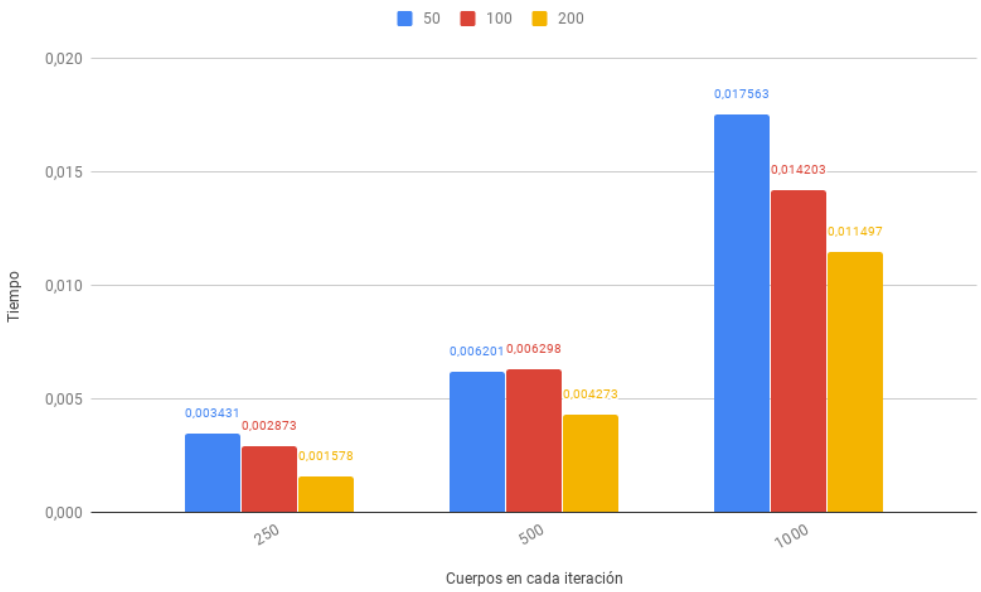
2.3.5. 16 hilos

Tiempo medio por ejecución



Asteroides Iteraciones Planetas → Tiempo
250 50 250 → 0,182035 s
250 100 250 → 0,279373 s
250 200 250 → 0,296924 s
500 50 500 → 0,311823 s
500 100 500 → 0,609455 s
500 200 500 → 0,824987 s
1000 50 1000 → 0,872044 s
1000 100 1000 → 1,398234 s
1000 200 1000 → 2,242337 s

Tiempo medio por iteración



Asteroides Iteraciones Planetas → Tiempo
250 50 250 → 0,003431 s
250 100 250 → 0,002873 s
250 200 250 → 0,001578 s
500 50 500 → 0,006201 s
500 100 500 → 0,006298 s
500 200 500 → 0,004273 s
1000 50 1000 → 0,017563 s
1000 100 1000 → 0,014203 s
1000 200 1000 → 0,011497 s

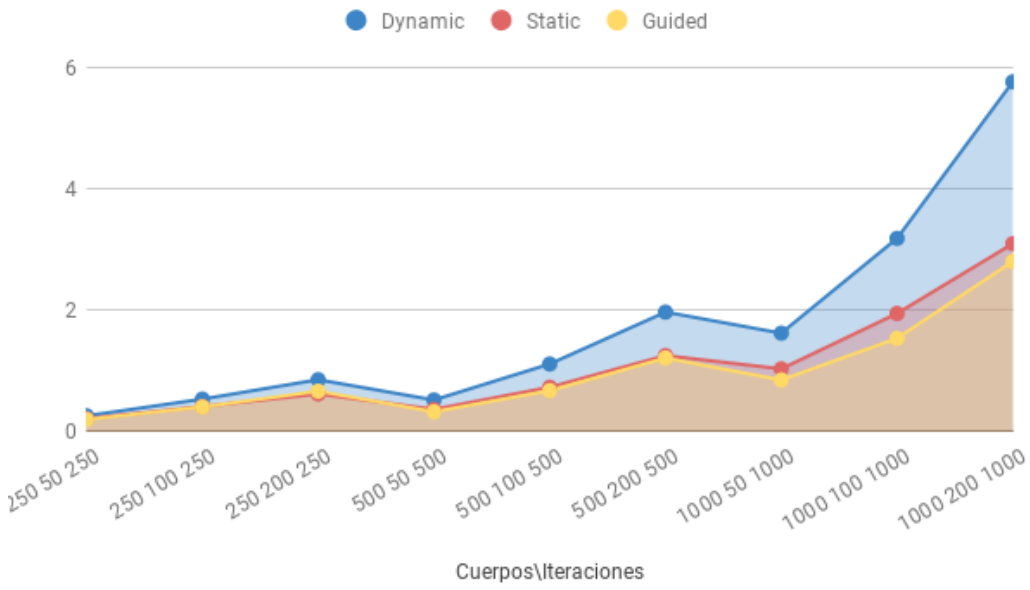
2.4. Análisis de resultados

Para analizar los resultados, hemos calculado la media de los resultados obtenidos por número de hilos, y los hemos comparado con la media de los resultados obtenidos en la secuencial. De esta manera hemos calculado los siguientes speedups:

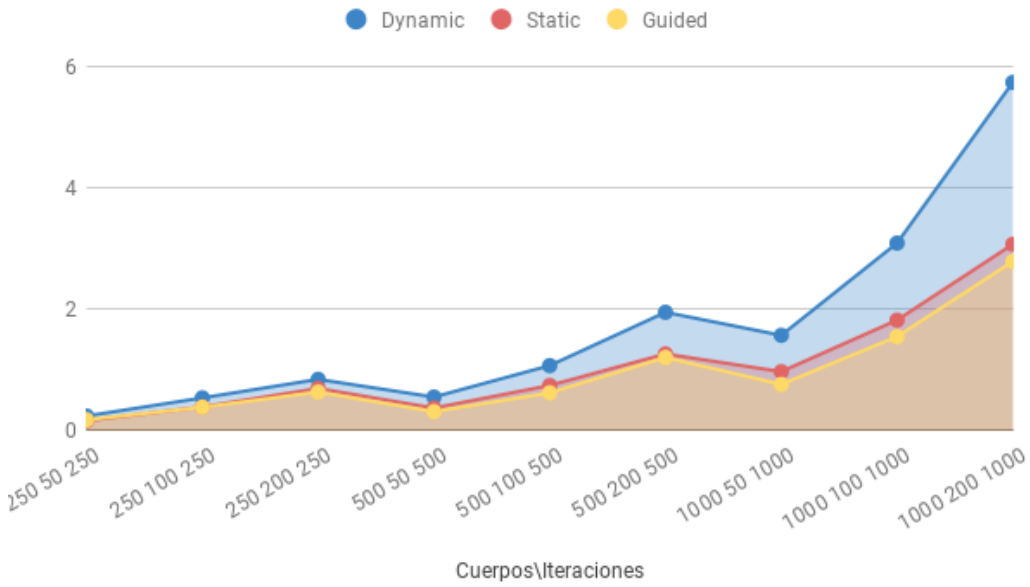
Hilos	Speedup ejecución	Speedup por iteración
2	2.685	2.574
4	2.704	2.515
8	2.751	2.573
16	2.72	2.618

2.5. Impacto de la planificación

2.5.1. 4 Hilos



2.5.2. 8 Hilos



3. Plan de pruebas

3.1. Versión secuencial

Descripción de la Prueba	Resultado esperado
Introducimos un número de argumentos inferior a 4	Notificación de error
Introducimos un número de argumentos superior a 4	Notifiación de error
Introducimos 4 argumentos y comprobamos la primera línea del fichero inicial	Mismos números que los argumentos introducidos
Comparación del fichero inicial con el generado por el binario proporcionado	Mismos resultado
Contar el número de lineas en el fichero de salida	Mismas lineas que asteroides pasados por argumento
Comparación del fichero de salida con el generado por el binario proporcionado	Margen de error de ± 5 en la posición (tanto x como y)

3.2. Versión paralela

Descripción de la Prueba	Resultado esperado
Mismas pruebas que secuencial	Mismos resultados
Comparación fichero de salida versión secuencial y paralela	Resultados idénticos

4. Conclusión

La librería OpenMP facilita mucho la implementación de paralelismo, ya que con una sintaxis compacta y sencilla, se pueden paralelizar facilmente los programas, utilizando secciones y el paralelismo de los bucles for. Además, después de haber realizado paralelismo a través de —pthreads— en la asignatura de sistemas operativos de segundo, valoramos más la fácil implementación de esta librería. Añadir que aunque el paralelismo es notable en todos los casos, en este concreto es más significativo cuanta mayor cantidad de asteroides y planetas haya, y el número de iteraciones no influye en el tiempo ahorrado.