INGENIERÍA DE LOS COMPUTADORES

PRÁCTICAS 1 y 2

Introducción a los problemas paralelos y proyecto de evaluación paralela

Francisco José Castellanos
Javier Ortega Bastida
Antonio Miguel Rodríguez
Ismael Piñeiro Ramos

Índice

1. Objetivos de la Práctica

Esta primera práctica, estará dividida en dos partes: en la primera parte se pretende que los miembros del grupo tomen contacto con la problemática asociada a las arquitecturas de altas prestaciones, respondiendo, con ejemplos concretos, a las siguientes preguntas: “¿para qué sirve un multicomputador?” y “¿para qué sirve el paralelismo en computación?”.

Resolviendo estas cuestiones, los integrantes del grupo, podrán obtener información suficiente y actualizada sobre conceptos de gran difusión relacionado con las arquitecturas de altas prestaciones y el desarrollo de problemas paralelos.

Una vez respondidas estas preguntas, se podrá abordar la segunda parte de la práctica, que consistirá en plantear un problema con elevadas necesidades computacionales que posteriormente mejoraremos en futuras prácticas y deberá ser un problema real del mundo de la ingeniería.

Respecto al problema, se deberá implementar una solución tradicional (sin hacer uso del paralelismo). De esta forma, se podrá apreciar las dificultades que ofrece un programa de altas prestaciones computacionales desarrollado de forma tradicional.

2. Introducción a los Sistemas Paralelos

En la práctica se pretende introducir a los conceptos básicos del paralelismo para entender la ventaja que nos ofrece para la resolución de problemas con altas prestaciones de cómputo. En primer lugar, debemos entender qué es el paralelismo.

El paralelismo es una acción que realiza el procesador para ejecutar varias tareas al mismo tiempo. Es decir, puede realizar varios cálculos simultáneamente, basado en el principio de dividir los problemas grandes para obtener varios problemas pequeños, que son posteriormente solucionados en paralelo, consiguiendo así, reducir el tiempo de ejecución de nuestras aplicaciones.

Hay distintos tipos de paralelismo:

- **Nivel de bit**: Se basa en el tamaño de la palabra que es capaz de manejar el procesador (8, 16, 32, 64, etc.), mientras más grande el tamaño de la palabra menos instrucciones ejecuta el procesador para realizar una operación determinada.

- **Nivel de instrucción**: Los mecanismos de la arquitectura son utilizados entonces para ejecutar este tipo de paralelismo (en las lecciones de teoría de la asignatura se han explicado los superescalares). Es la capacidad de procesar instrucciones en paralelo, determinada por el número de instrucciones que pueden solaparse en las etapas de un procesador.

- **Nivel de datos**: Este tipo de paralelismo se enfoca en la distribución de los datos entre varios procesadores. Se conoce también como paralelismo a nivel de lazos.

- **Nivel tarea**: En este caso un programa paralelo que ejecuta cálculos distintos sobre el mismo conjunto de datos o sobre datos diferentes.

Resumiendo, el paralelismo permite reducir la problemática asociada a las arquitecturas de altas prestaciones para obtener mejores tiempos de ejecución. Para entender de forma más clara los conceptos anteriores, vamos a resolver, con ejemplos aclaratorios, las siguientes cuestiones.

¿Para qué sirve un multicomputador?

Un multicomputador, es un sistema formado por un conjunto de computadores, que habitualmente poseen similares capacidades y características de rendimiento.

Se utiliza para que las tareas se puedan distribuir entre los diferentes ordenadores que integran el conjunto del multicomputador y hacer el trabajo más rápido.

Ejemplos de tareas que se pueden mejorar con un multicomputador: una productora de cine de animación por ordenador, una escena se podría dividir en pequeños trozos, y cada uno de ellos pasaría a ser tratado por un ordenador distinto.

Por otra parte, con sistemas multicomputador se pueden llegar a atender peticiones de miles de usuarios de la misma manera que los grandes servicios de búsqueda de Internet disponen de decenas de miles de ordenadores para atender los millones de peticiones que reciben cada minuto.

¿Para qué sirve el paralelismo en computación?

La necesidad de realizar unos cálculos más complejos llevó al desarrollo del paralelismo. El paralelismo, basada sobre todo en la división del trabajo, es un método más económico y eficiente en comparación a realizar un aumento de la capacidad un procesador.

En definitiva, el paralelismo sirve para resolver problemas considerados anteriormente muy largos y costosos que incluso no se hayan podido solucionar hasta el momento.

Un ejemplo real de la utilidad del paralelismo podría ser: en el mundo de la economía, la matemática financiera, que consiste en gran cantidad de cálculos complejos relacionados con operaciones financieras.

En el mundo de la informática, es útil para los programas que utilizan la programación dinámica, un método para reducir el tiempo de ejecución de un algoritmo mediante la utilización de subproblemas superpuestos y subestructuras óptimas y en otros programas que se resuelven empleando el algoritmo de ramificación y poda.

3. Descripción del problema a resolver

El problema que vamos a implementar es una adaptación del problema conocido como “problema de los n-cuerpos”. Existen varias variaciones de este problema como pueden ser el problema de los dos o tres cuerpos, el cual consiste en dado tres cuerpos determinar en qué posición se encontraran en un instante de tiempo, estando sometidos a distintas fuerzas gravitatorias u otras magnitudes físicas.

El problema de los n-cuerpos consiste en buscar ecuaciones de movimiento adecuadas para un número variable de cuerpos (partículas) en interacción gravitacional. Este tipo de problemas es muy usado en distintas áreas, cabe destacar su gran uso en la ciencia de la astrofísica para estudiar el movimiento de los cuerpos celestes, la influencia de los agujeros negros y cosmología.

Un claro ejemplo de este problema puede ser nuestro sistema solar, donde la posición de los planetas al moverse alrededor del sol resulta en una variación de la fuerza ejercida sobre alguno de sus miembros.

La adaptación del problema que vamos a realizar consiste en una aplicación la cual calculará la posición de un número variable de cuerpos en un sistema de 2D teniendo una fuerza de atracción (gravedad) en el centro del escenario y los cuerpos una velocidad ambas constantes. Por defecto el escenario se traduce en un tablero cuadrado de 1000x1000.

4. Planteamiento de la solución

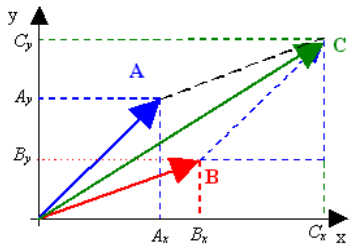
Para esta práctica hemos decidido realizar 2 programas. Un programa principal diseñado en C que es el que calculará las posiciones de los cuerpos para cada instante de tiempo. El otro programa está implementado en Java utilizando la librería Canvas y nos ayudará a comprobar si las posiciones calculadas por el programa en C son correctas o no. El programa de Java mostrará un tablero en 2D en el que se podrá observar el movimiento de los cuerpos alrededor del centro de gravedad fijo.

El programa principal pide al usuario una serie de datos necesarios para comenzar como son:

- Número de cuerpos
- Velocidad inicial de los cuerpos (la misma para todos)
- Posiciones iniciales de los cuerpos
- Número de movimientos a realizar (instantes de tiempo)
- Fuerza de la gravedad (situada en el centro)

Una vez se tienen estos datos, realiza los cálculos por cada instante de tiempo que llamamos movimiento. La idea en futuras prácticas es paralelizar los cálculos de las diferentes coordenadas x e y de los diferentes cuerpos para así poder realizar el cálculo de un número indeterminado de cuerpos y de movimientos en un menor tiempo.

El cálculo de la posición de los cuerpos se basa en la suma de vectores y en la trigonometría. El vector de velocidad del cuerpo ya lo tenemos desde el principio. En cada instante de tiempo se modifica dicha velocidad según si la gravedad ejerce aceleración o desaceleración al mismo. Lo único que debemos hacer es obtener el vector de gravedad para cada movimiento, y calcular mediante trigonometría (seno y coseno) la aceleración que aporta tanto en el eje X como en el eje Y, y teniendo en cuenta la fuerza de gravedad que habíamos introducido por parámetros. Después de saber la nueva velocidad para los dos ejes, cambiamos el vector de velocidad del cuerpo por su nuevo vector. EL ángulo de incidencia de la gravedad respecto al vector de velocidad del cuerpo se obtiene mediante la ecuación punto pendiente.



$$m = \frac{y_2 - y_1}{x_2 - x_1} \qquad m = tg\ \alpha$$

El programa tiene dos modos de ejecución:

1. Si ejecutamos la aplicación añadiendo un parámetro cualquiera, se generará un fichero de texto plano en el cual se reflejarán los distintos movimientos realizados, además de los parámetros de configuración de ese determinado problema.

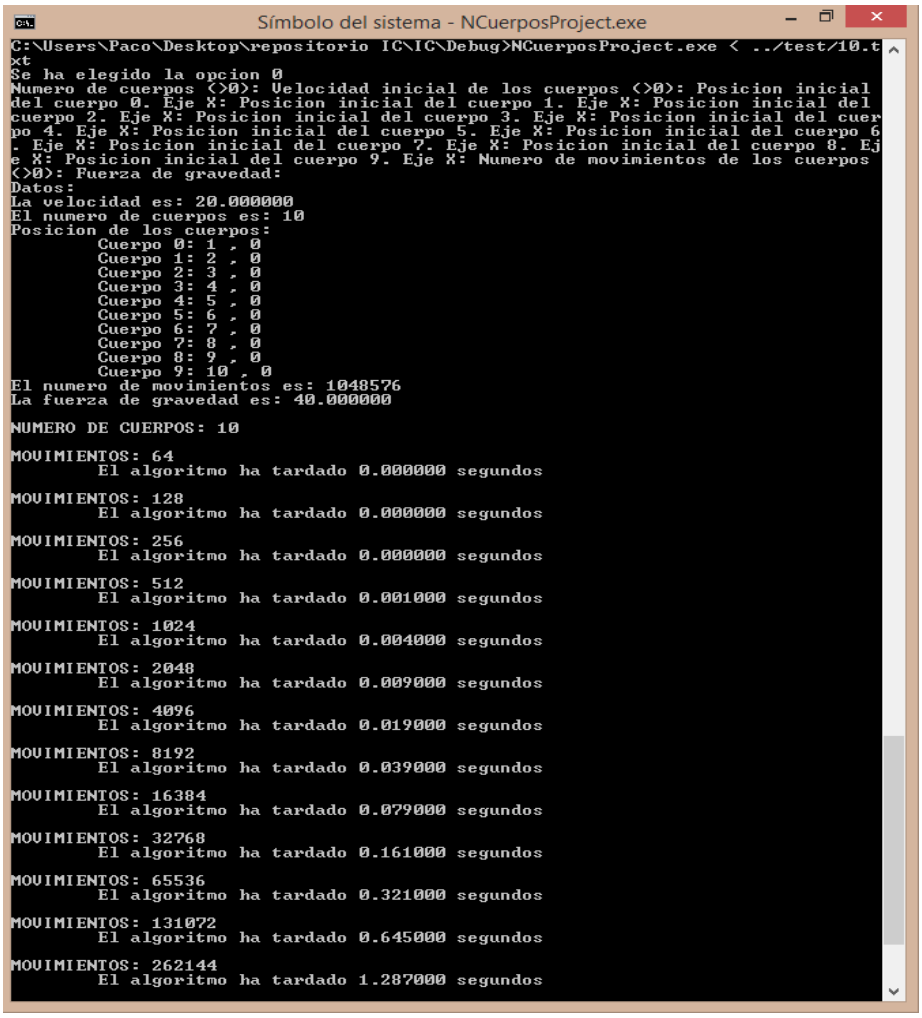
2. Si ejecutamos la aplicación sin parámetros, el programa no genera ficheros, y realiza un conjunto de pruebas con diferente número de movimientos para el número de cuerpos que el usuario ha indicado. Por simplificar, el programa pide exactamente los mismos datos que el modo anterior, aunque realmente el número de movimientos lo obvia. Las pruebas que realiza comienzan con 64 movimientos, y aumenta en potencias de 2 hasta más de 8 millones de movimientos, más que suficiente para obtener resultados interesantes.

El objeto de obtener un fichero de texto plano en el primer modo de ejecución es, además de visualizar y analizar los resultados, poder leer este fichero con el otro programa escrito en Java (por facilidad a la hora de crear interfaces gráficas) y mostrar los resultados en un pequeño tablero de 2D para hacer más atractiva la solución del problema mediante la librería Canvas. A continuación se muestra el aspecto del tablero.

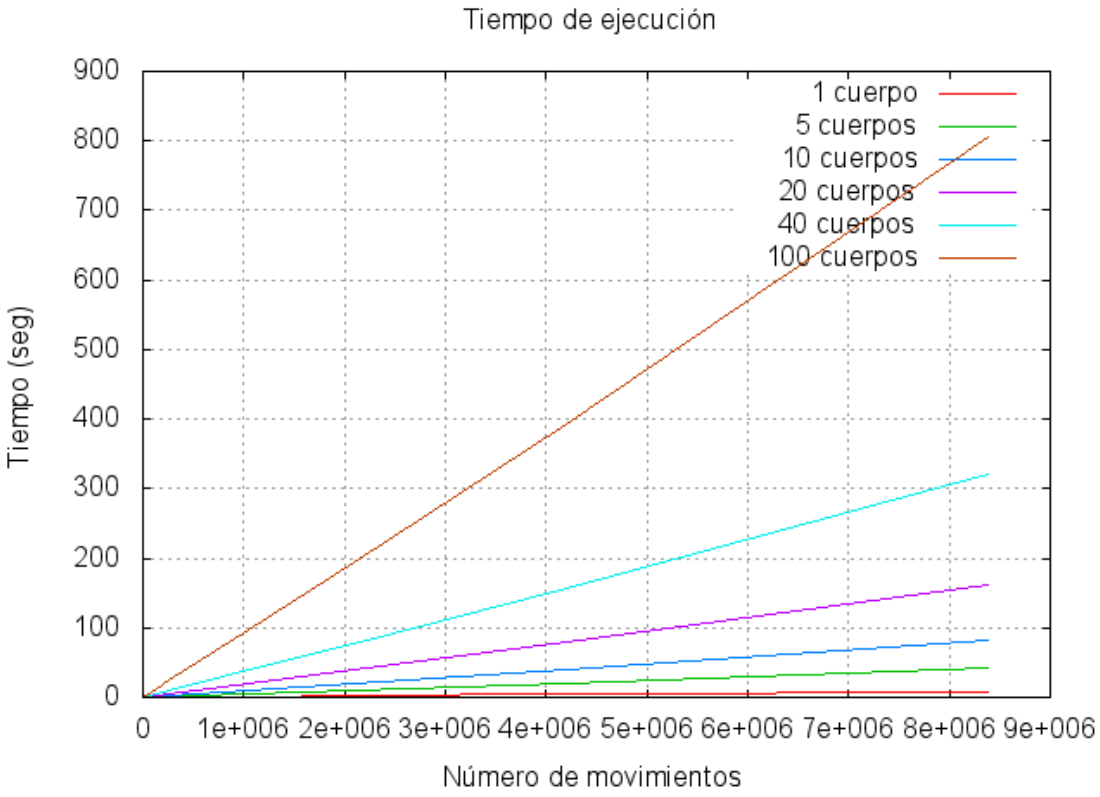
Siendo el cuadrado rojo el punto estático que representa el centro de gravedad y el resto de cuadrados los cuerpos.

5. Evaluación de resultados

A razón de comparar el algoritmo actual con el de futuras prácticas analizamos los tiempos de ejecución para distinta carga. A continuación podemos ver un ejemplo de ejecución del programa. Observamos que el programa permite realizar multitud de ejecuciones con distintas configuraciones para posteriormente estudiar los resultados y compararlos con otras ejecuciones. Nosotros elegimos el número de cuerpos que se van a generar en la prueba, y el programa calcula el tiempo que tarda en calcular las posiciones de los mismos en cada instante de tiempo para un diferente número de movimientos o iteraciones. Está diseñado para variar el número de movimientos en un factor de potencias de 2 a partir del valor 64.



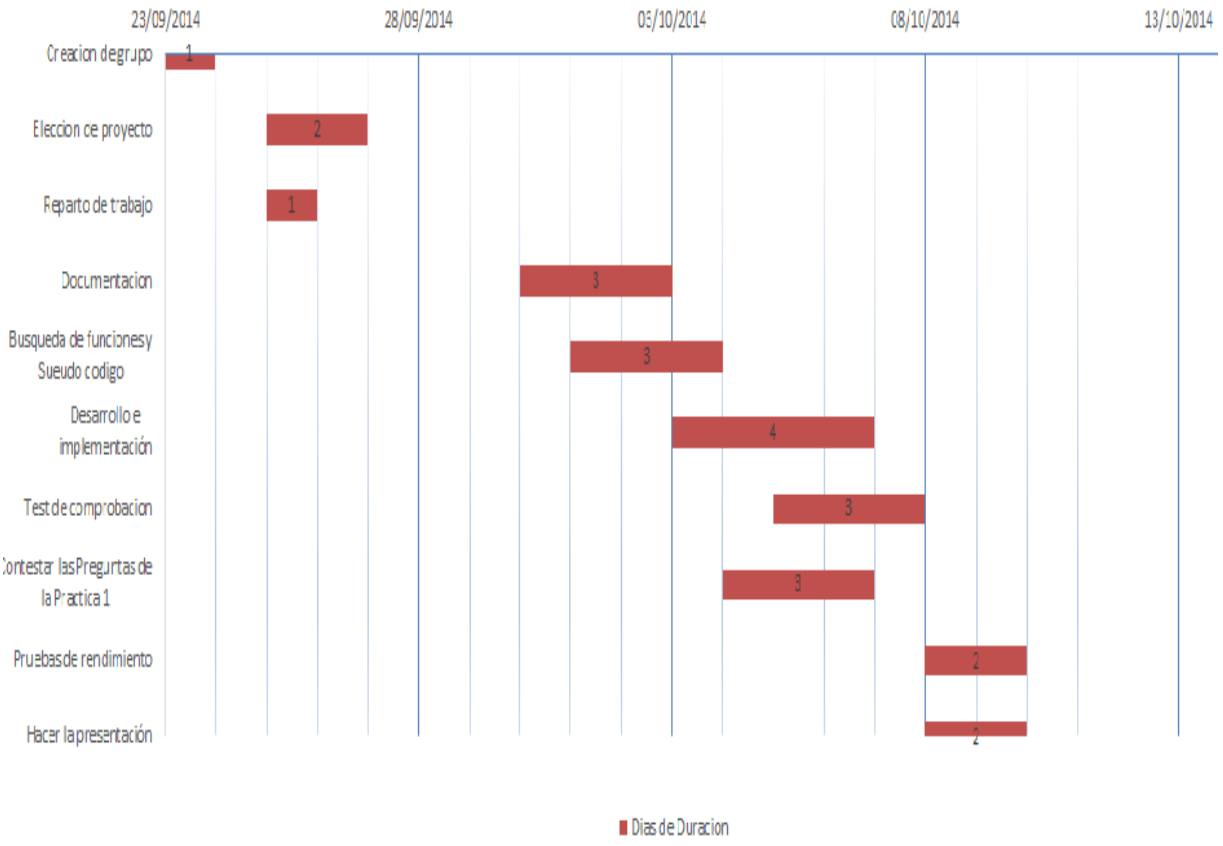
Tras realizar la ejecución del programa para diferentes números de cuerpos, hemos confeccionado una gráfica que muestra los resultados de manera que podamos comparar los tiempos de ejecución según el número de cuerpos y de movimientos. En este caso hemos realizado pruebas con 1, 5, 10, 20, 40 y 100 cuerpos.



Observamos una tendencia a aumentar de manera prácticamente lineal en todos los casos. El número de cuerpos es un factor muy a tener en cuenta a la hora de ejecutar este programa, ya que podemos considerar que a partir de la prueba de 40 cuerpos, la pendiente de la recta empieza a aumentar bastante y que para un número de movimientos mayor de 8 millones, se obtiene un tiempo de ejecución alrededor de 5 minutos, siendo necesaria una optimización en el código, como utilizar el paralelismo sobre todo a mayor número de cuerpos.

6. Plan de trabajo, diagrama GANTT

A continuación, mostramos el diagrama de Gantt con las semanas de trabajo que llevó el desarrollo de cada tarea de esta práctica.



7. Conclusiones

Tras las pruebas realizadas, deducimos que hay casos en que un programa diseñado de manera secuencial puede ser muy poco eficiente en cuanto a tiempo de ejecución. Nos damos cuenta de que es necesario optimizar los resultados de alguna manera, siendo el paralelismo un posible candidato para obtener una mejora sustancial de los resultados, ya que se trata de una herramienta útil y a día de hoy necesaria para el desarrollo de problemas que necesitan una alta densidad de cálculos.

8. BIBLIOGRAFÍA

Paralelismo en informática

- http://www.ecured.cu/index.php/Paralelismo_%28_inform%C3%A1tica%29

Multicomputadores

- http://edukanda.es/mediatecaweb/data/zip/638/PID_00150197/web/main/m1/v4_4.html

Operaciones con vectores

- http://hyperphysics.phy-astr.gsu.edu/hbasees/vect.html

Problema de los 3 cuerpos

- http://es.wikipedia.org/wiki/Problema_de_los_tres_cuerpos

Problema de los N cuerpos

- http://www.ugr.es/~isa/numerical-methods2.pdf

- https://freeshell.de/~rgh/arch/mec-unsj-teo-4.pdf