Programación orientada al rendimiento: Proyecto de programación paralela

J. Daniel García Sánchez (coordinador)

Arquitectura de Computadores
Departamento de Informática
Universidad Carlos III de Madrid

1. Objetivo

Este proyecto tiene como objetivo fundamental hacer que los estudiantes adquieran **preocupación por el rendimiento de programas paralelos** y se familiaricen con las **técnicas de programación paralela**. Así mismo, se introducirán las **técnicas de evaluación del rendimiento en programas paralelos**.

En concreto, la práctica se centrará en el desarrollo de software paralelo en el lenguaje de programación C++ (incluyendo sus últimas versiones), a partir de un software secuencial desarrollado previamente.

2. Descripción del proyecto

Este proyecto consiste en la generación de las versiones paralelas de la aplicación de simulación gravitatoria de objetos realizada en la primera práctica. Para la paralelización del código se utilizará **OpenMP**.

Se implementarán dos versiones paralelas del programa utilizando las estrategias: arrays de estructuras/objetos (**paos**) y estructuras/objetos de arrays (**psoa**).

El programa debe producir exactamente los mismos resultados que la versión secuencial de las aplicaciones.

3. Tareas

3.1. Desarrollo de versión paralela

Esta tarea consiste en el desarrollo de la versión paralela de la aplicación descrita en C++17 (o C++20).

3.1.1. Configuración del compilador

Todos sus archivos fuente deben compilar sin problemas y no deben emitir ninguna advertencia del compilador. En particular, deberá activar, al menos, los siguientes flags del compilador relativos a *warnings*:

- `-Wall -Wextra -Wno-deprecated -Werror -pedantic -pedantic-errors`,

Tenga también en cuenta que deberá realizar todas las evaluaciones con las optimizaciones del compilador activadas (opciones del compilador `-O3` y `-DNDEBUG`). Puede conseguir esto de una forma sencilla con `-DCMAKE_BUILD_TYPE=Release`.

Se podrán activar flags adicionales de optimización siempre que se documente en la memoria del proyecto y se justifique la utilización de los mismos. Debe incluir dichos flags en su archivo de configuración de CMake.

3.1.2. Bibliotecas

Está permitido el uso de cualquier componente de la biblioteca estándar de C++ que está incluida en la distribución del compilador. Esto incluye muchos archivos de cabecera como **<vector>**, **<list>**, **<fstream>**, **<cmath>**, **<random>**, ...

También está permitido el uso de la biblioteca de **OpenMP** (archivo de cabecera **<omp.h>**).

En general, no se permiten bibliotecas adicionales. No obstante, el coordinador de la asignatura podrá conceder permisos a aquellos estudiantes que lo soliciten de manera justificada.

3.2. Evaluación del rendimiento

Esta tarea consiste en evaluar el rendimiento de la aplicación paralela en sus dos versiones (**paos** y **psoa**).

Para evaluar el rendimiento debe medir el tiempo de ejecución de la aplicación. Debe representar gráficamente los resultados. Tenga en cuenta las siguientes consideraciones:

- Todas las evaluaciones deben realizarse en la misma máquina utilizada para el primer proyecto de la asignatura.
- Debe incluir en la memoria todos los parámetros relevantes de la máquina en la que ha ejecutado (modelo de procesador, número de cores físicos, tamaño de memoria principal, jerarquía de la memoria caché, ...) y del software de sistema (versión de sistema operativo, versión del compilador, ...).
    - Se deberá utilizar un computador que disponga de un procesador con, al menos, cuatro núcleos físicos.
    - El sistema operativo deberá ser GNU/Linux instalado de forma nativa (no se admite el uso de una máquina virtual).
    - El compilador deberá ser **gcc** versión 9 o superior.
- Realice cada experimento un número de veces y tome el valor promedio. Se recomienda un mínimo de 10 o más ejecuciones por experimento y que proporcione un intervalo de confianza.
- Estudie los resultados para varios tamaños de la población de objetos. Considere los casos de 4000 y 8000.
- Estudie los resultados para distintos números de iteraciones: 250 y 500.

Represente en una gráfica todos los tiempos totales de ejecución obtenidos. Represente en otra gráfica el tiempo medio por iteración.

Debe realizar la evaluación del rendimiento considerando un número distintos de hilos de ejecución, variando desde 1 hasta 16 hilos (1, 2, 4, 8 y 16). Deberá representar el speedup con respecto de la versión secuencial.

Incluya en la memoria de esta práctica las conclusiones que pueda inferir de los resultados. No se limite simplemente a describir los datos. Debe buscar también una explicación convincente de los resultados que incluya el impacto de la arquitectura del computador.

4. Calificación

La puntuación final obtenida en este proyecto se obtiene teniendo en cuenta el siguiente reparto:

- **Rendimiento alcanzado** (70 %).
- **Calidad de la memoria** (30 %).

Advertencias:

- Si el código entregado no compila, la nota final de la práctica será de 0.
- En caso de copia todos los grupos implicados obtendrán una nota de 0.

5. Procedimiento de entrega

La entrega del código y de la correspondiente memoria del proyecto se realizará a través de Aula Global.

- **Entrega del código fuente**
  - Se entregará un archivo comprimido en formato **zip** que contendrá todo el código fuente de la aplicación.
  - Deben incluirse todos los archivos **.hpp** y **.cpp** utilizados.
  - También deben incluirse todos los archivos CMake (**CMakeLists.txt**) para generar el proceso de compilación.
  - No debe incluirse ningún archivo resultado de la compilación como archivos objeto, bibliotecas en formato binario o ejecutables.
- **Memoria del proyecto**
  - Debe entregarse una memoria en formato PDF.
  - Evite incluir en la memoria código fuente. Si fuese necesario haga referencia al código fuente entregado.
  - La memoria deberá incluir los siguientes contenidos:
    - **Página de título**. Contendrá:
      - El nombre del proyecto.
      - El grupo reducido en el que están matriculados los estudiantes.
      - El número de equipo asignado.
      - El nombre y NIA de los autores.
  - **Índice de contenidos**.

- **Decisiones de diseño**. Explicará las decisiones de diseño adoptadas en el proceso de paralelización. También se incluirán las alternativas de diseño utilizadas y las principales modificaciones realizadas.
- **Implementación**. Incluirá una discusión de los aspectos de implementación de las decisiones de diseño y su impacto. Opcionalmente, podrá incluir (si fuese el caso) la utilización de flags adicionales de optimización explicando sus ventajas e inconvenientes.
- **Evaluación de rendimiento**. Presentará las diversas evaluaciones del rendimiento realizadas comparando la versión original secuencial entregada en el proyecto anterior y la versión paralelizada.
- **Conclusiones**. Se valorarán especialmente las derivadas de los resultados de la evaluación del rendimiento, así como las que relacionen el trabajo realizado con el contenido de la asignatura.

La memoria no deberá sobrepasar las 15 páginas incluyendo la portada y todas las secciones. No se tendrá en cuenta en la corrección los contenidos a partir de la página 16 si fuese el caso.