Práctica de programación paralela con OpenMP

J. Daniel García Sánchez (coordinador)
David Expósito Singh
Javier García Blas
Óscar Pérez Alonso
J. Manuel Pérez Lobato

Arquitectura de Computadores
Grupo ARCOS
Departamento de Informática
Universidad Carlos III de Madrid

1. Objetivo

Esta práctica tiene como objetivo fundamental familiarizar al estudiante con la programación paralela en arquitecturas de memoria compartida. En concreto, la práctica se centrará en la familiarización con el modelo de programación de OpenMP.

2. Descripción de la práctica

Para realizar esta práctica se suministran dos implementaciones de la solución al problema de los N-cuerpos escritas en C++. Las dos implementaciones suministrados abordan el problema con las estrategias AOS (Array of Structures) y SOA (Structure of Arrays).

2.1. El problema de N-cuerpos

El problema a resolver consiste en simular a lo largo del tiempo el movimiento de N cuerpos teniendo en cuenta la atracción gravitatoria. Este cálculo se realiza a lo largo del tiempo cada cierto incremento de tiempo Δt.

El programa realiza la simulación en un espacio bidimensional. Se considera que el espacio es un recinto cerrado. De esta manera cuando un objeto impacta con el borde del recinto, se produce un rebote, cambiando la dirección del movimiento del objeto.

2.2. Código suministrado

El código suministrado está formado por un conjunto de clases y funciones de biblioteca y dos programas principales.

Los programas principales se encuentran en el directorio *app*. El resto de ficheros fuente se encuentran en los directorios *include* y *src*.

El programa **nbodyfile** lee de un fichero la información de los parámetros de simulación y el estado inicial de los objetos a simular. Dicho fichero contiene una línea de cabecera con los parámetros de

simulación, seguido de un conjunto de líneas (una por objeto) que contiene el estado inicial de cada objeto. Toma los siguientes parámetros en la línea de comandos:

- **modo**: Modo de ejecución. Puede ser *aos* (*array of structures*) o *soa* (*structure of arrays*).
- Nombre del fichero de entrada. Si no se especifica, se usará el fichero `in.txt`.
- Nombre del fichero de salida. Si no se especifica, se usará el fichero `out.txt`.

El programa **nbodyrnd** genera un conjunto de objetos de forma pseudoaleatoria a partir de dos parámetros: el número de objetos y el número de iteraciones de la simulación. Este programa genera un cierto número de objetos que se distribuyen de forma uniforme en el espacio y cuya masa se genera siguiendo una distribución normal. El programa toma los siguientes parámetros en la línea de comandos:

- **modo**: Modo de ejecución. Puede ser *aos* (*array of structures*) o *soa* (*structure of arrays*).
- Número de cuerpos a simular. Si no se especifica se usan 50 cuerpos.
- Número de iteraciones para la simulación. Si no se especifica se usan 100 iteraciones.
- Nombre del fichero de salida. Si no se especifica, se usará el fichero `out.txt`.

3. Tareas

3.1. Evaluación inicial del rendimiento

Esta tarea consiste en evaluar el rendimiento inicial de los programas suministrados. Puede usar el programa *nbodyfile* para experimentar con el código y comprender su funcionamiento. Posteriormente ejecute el programa *nbodyrnd* con distintos parámetros de entrada.

Nota importante: Asegúrese de compilar todos los ejemplos con las optimizaciones activadas (opcion del compilador `-O3`, u opción de *CMake* `CMAKE_BUILD_TYPE=Release`).

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 las aulas de la universidad. Debe incluir en la memoria todos los parámetros relevantes de la máquina en la que ha ejecutado (modelo de procesador, número de cores, tamaño de memoria principal, jerarquía de la memoria caché, ...) y del software de sistema (versión de sistema operativo, versión del compilador, ...).
- Realice cada experimento un número de veces y tome el valor promedio. Se recomienda un mínimo de 10 ejecuciones por experimento.
- Estudie los resultados para tamaños de la población de objetos que vaya desde 250 objetos hasta 1,000 objetos en incrementos de 250 (250, 500, 750, 1,000).
- Estudie los resultados para distintos números de iteraciones que vayan desde 50 hasta 200 en incrementos de 50.
- Recuerde que el objetivo es comparar las estrategias *aos* Y *soa*.

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

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.

3.2. Paralelización

Esta tarea consiste en desarrollar las correspondientes versiones paralelas de los programas suministrados usando OpenMP. Deberá explicar en la memoria de forma detallada las modificaciones que ha realizado al código.

Tenga en cuenta lo siguiente:

- Cualquier práctica que emita un advertencia (*warning*) se considerará suspensa.
- Sus programas deben incluir entre los parámetros pasados al compilador, como mínimo, los siguientes: `-std=c++14 -Wall -Wextra -Wno-deprecated -Werror -pedantic -pedantic-errors`, o parámetros equivalentes.

En la memoria debe incluir las decisiones de diseño que ha tomada para alcanzar la paralelización. Para cada decisión debe incluir que otras alternativa se podían considerar y justificar de forma razonada los motivos de la elección realizada.

3.3. Evaluación de versión paralela

Debe repetir las evaluaciones del primer apartado. Considere distinto número de hilos de ejecución, variando desde 1 hasta 16 hilos (1, 2, 4, 8 y 16).

Nota importante: Asegúrese de compilar todos los ejemplos con las optimizaciones activadas (opcion del compilador `-O3`, u opción de *CMake* `CMAKE_BUILD_TYPE=Release`).

Además de las gráficas del primer apartado, represente de forma gráfica el *speedup*.

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.

3.4. Impacto de la planificación

Realice un estudio para los casos de 4 y de 16 hilos del impacto que tienen los distintos modelos de planificación (*static*, *dynamic* y *guided*) incluidos en OpenMP. Incluya en la memoria las conclusiones que pueda inferir de los resultados obtenidos.