UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Computación Paralela (CPA)

Práctica 2: Computación paralela con planetas

Curso 2020/21

Índice

Advertencia

La memoria y el código fuente a entregar en esta práctica deben ser el trabajo personal y original del correspondiente grupo (de un máximo de dos estudiantes, ambos pertenecientes al mismo grupo de prácticas). La copia de cualquier parte de la memoria o del código de cualquier otra fuente, supondrá una calificación de cero en el trabajo, al margen de las medidas disciplinarias que pudieran derivarse.

1. Código de partida

En esta práctica partimos de un código (fichero `planets.c`) que procesa datos de un conjunto de planetas. Dichos datos están formados por las posiciones (coordenadas 3D) y masas de los planetas, a partir de los cuales el programa calcula el centro de masas del conjunto, además de indicar cuál es el planeta que tiene más planetas vecinos. Se entiende que dos planetas son "vecinos" si la distancia entre ellos es inferior a una distancia determinada `NGB_DST`.

Este procesamiento se realiza en la función `process_data`. El primer bucle de la función (con variable de control `i`) recorre los planetas, acumulando la aportación de cada uno de ellos al centro de masas del conjunto y obteniendo el número de vecinos de cada planeta.

Para obtener el número de vecinos, se utiliza un bucle anidado con variable de control `j`. Cada iteración del bucle `j` calcula la distancia entre los planetas `i` y `j` y, si los planetas se consideran vecinos, incrementa el número de vecinos tanto del planeta `i` como del `j`. Observa que el bucle `j` empieza en `i+1` y no en `0`, porque los planetas `0` a `i-1` ya han sido tenidos en cuenta en las iteraciones anteriores del bucle `i`. Por ejemplo, si los planetas 4 y 15 son vecinos, en la iteración `i=4` se incrementará el número de vecinos tanto del planeta 4 como del 15.

Tras finalizar esos dos bucles anidados, se termina de calcular el centro de masas y se obtiene el planeta con mayor número de vecinos (mediante un segundo bucle `i`). Observa que si hay varios planetas con el máximo número de vecinos, el programa se queda con el de índice menor. La versión paralela debe comportarse igual.

El programa acepta como argumento el número de planetas a considerar (`n`), por ejemplo:

```
$ ./planets 5000
The center of mass is in ( 1180.98, 1184.01, 1175.36 ).
The planet with most neighbors is 569 (15 neighbors).
```

Si se omite el argumento, se usa `n=10000`.

Los datos de los planetas se generan aleatoriamente (en la función `generate_data`), aunque se parte de una semilla fija, lo que hace que los datos generados no varíen si el programa se ejecuta múltiples veces con el mismo número de planetas.

2. Trabajo a realizar

El trabajo a realizar consiste básicamente en paralelizar el código de diferentes maneras, y realizar ejecuciones para poder comparar las prestaciones obtenidas por las diferentes versiones.

Ejercicio 1: Haz una versión (en un fichero con nombre `planets1.c`) en la que se muestre el tiempo de cálculo del programa (el tiempo empleado por la función `process_data`) y también el número de planetas (`n`). Aunque el programa sea secuencial, utiliza la función de OpenMP apropiada para tomar tiempos.

Esto servirá para poder comparar el tiempo con los programas paralelos.

Ejercicio 2: Paraleliza los cálculos que se realizan en la función `process_data`. Debes hacer dos versiones diferentes: una (`planets2i.c`) en la que se paralelice el primer bucle `i` (bucle externo) y otra (`planets2j.c`) en la que se paralelice el bucle `j` (interno). En ambas versiones debes paralelizar también el segundo bucle `i`, así como hacer que el programa muestre, junto al tiempo de ejecución y número de planetas, el número de hilos utilizado. En caso de que alguno de los tres bucles no se pueda paralelizar, explica por qué y no lo paralelices.

En ambas paralelizaciones deberás prestar especial atención a la actualización del número de vecinos. Considera el bucle que estás paralelizando y piensa si dos iteraciones de ese bucle pueden estar modificando el número de vecinos de un mismo planeta.

Comprueba que las paralelizaciones realizadas funcionan correctamente.

Ejercicio 3: Para cada bucle paralelizado, ¿crees que la planificación utilizada afectará a las prestaciones obtenidas? Razona la respuesta.

Ejercicio 4: Utilizando el cluster kahan, saca tiempos de las paralelizaciones realizadas, usando diferentes planificaciones en caso de que creas, según lo respondido en el ejercicio anterior, que la planificación puede afectar. Utiliza un número de planetas que consiga que el tiempo secuencial esté entre 30 y 90 segundos.

Para limitar el número de ejecuciones, se recomienda usar potencias de 2 para los valores del número de hilos (2, 4, 8...), llegando hasta el número de hilos que consideres adecuado (justifica por qué eliges ese número máximo de hilos).

Muestra tablas y gráficas para tiempos, speed-ups y eficiencias para las versiones paralelas que hayas realizado. En concreto, si has probado diferentes planificaciones de alguna versión, compara (mediante tablas y gráficas) las prestaciones de dichas planificaciones. Extrae conclusiones sobre si hay diferencias significativas entre las planificaciones y, en tal caso, qué planificaciones son mejores. Razona a qué crees que se debe.

Compara, de nuevo mediante tablas y gráficas, las prestaciones de las dos versiones paralelas, eligiendo la mejor planificación de una versión en caso de que hayas probado diferentes planificaciones de ella. En vista de los resultados, extrae conclusiones sobre cuál es la mejor versión paralela, o bien si no hay diferencia significativa, y razona a qué crees que se debe.

Explica cómo has lanzado las ejecuciones en el cluster, indicando cómo estableces el número de hilos y (en su caso) la planificación y adjuntado alguno de los ficheros de trabajo utilizados.

Ejercicio 5: Haz una versión (`planets5.c`) basada en `planets2i.c`, en la que se añada/modifique lo necesario para que cada hilo muestre por pantalla cuántos planetas le ha tocado procesar (número de iteraciones del primer bucle `i` que ha realizado) y el centro de masas de estos planetas. Por lo demás, el programa debe seguir funcionando como antes. No se pide analizar las prestaciones de esta versión.

3. Entrega

Hay **dos tareas** en PoliformaT para la entrega de esta práctica:

- En una de las tareas debes subir un fichero **en formato PDF** con la memoria de la práctica. No se admitirán otros formatos.
- En la otra tarea debes subir un único archivo comprimido con los ficheros de código fuente de las distintas versiones que hayas desarrollado, junto con alguno de los ficheros de trabajo utilizados para lanzar las ejecuciones. **No incluyas los ejecutables resultantes de la compilación** (ocupan mucho y no son necesarios porque se pueden generar a partir del fuente). El archivo debe estar comprimido en formato `.tgz` o `.zip`.

  Comprueba que todos los ficheros compilen correctamente y que tienen el nombre que se especifica en este boletín.

A la hora de realizar la entrega de la práctica, hay que tener en cuenta las siguientes recomendaciones:

- Hay que entregar una memoria descriptiva de los códigos empleados y los resultados obtenidos. Procura que la memoria tenga un tamaño razonable (ni un par de páginas, ni varias decenas).
- Los ejercicios 1-4 son obligatorios. El ejercicio 5 es opcional, aunque deberá entregarse si se desea poder optar al 100 % de la nota.
- No incluyas el código fuente completo de los programas en la memoria. Sí puedes incluir, si así lo deseas, las porciones de código que hayas modificado.
- Pon especial cuidado en preparar una buena memoria del trabajo. Se trata de entregar una memoria. No queremos un libro, pero sí que tenga una estructura y que tenga algo de narrativa y no una mera exposición de resultados.