Actividad

Actividad AE2_T2_Multiproceso

Ubicación

Tema 2 - Programación multiproceso

Objetivos

- Comprender el concepto de multiproceso.
- Saber crear múltiples procesos a partir de una aplicación Java.
- Aprender a gestionar la comunicación entre procesos.
- Desarrollar aplicaciones multiproceso para resolución de problemas complejos.

Temporalización

La duración prevista para esta actividad es de seis sesiones lectivas.

Instrucciones

La última orden presidencial de Donald Trump antes de finalizar su mandato fue decretar que la Tierra es plana. Al tener mayoría en el Congreso y en el Senado se aprobó sin mayores problemas y hoy en día el planeta es oficialmente plano en Estados Unidos. Una de las consecuencias es que la NASA ha dejado de recibir fondos y uno de sus departamentos, el Departamento de NEOs (Near-Earth Objects), se ha desmantelado por completo y ya no tienen ni personal ni equipamiento. Este departamento era el encargado de detectar posibles colisiones entre los asteroides y la Tierra. Desesperados por seguir con su labor, han lanzado una petición mundial para que desarrolladores de software de todo el mundo les ayuden a seguir con su labor. Nosotros vamos aportar nuestro grano de arena y ayudaremos implementando su algoritmo de detección en nuestro modesto PC. A continuación se expone lo que piden…

El objetivo del programa NEO Help! es que desarrolles una aplicación Java que calcule la probabilidad de que un objeto tipo NEO colisione con la Tierra en los próximos 50 años. La aplicación deberá realizar lo siguiente:

- Leer la información de un NEO (nombre, posicionNEO -posición relativa a la tierra- y velocidad NEO -velocidad en kilómetros por segundo relativa al Sol-) de un fichero de datos (se proporciona junto con este enunciado), en el que cada línea corresponde a un NEO, con los parámetros separados por comas.
- Calcular la probabilidad de que el NEO colisione con la Tierra.
- Guardar la probabilidad de colisión de cada NEO en un fichero independiente que se denomine <Nombre del NEO>.
- Mostrar como salida las probabilidades de colisión de cada NEO (con 2 decimales) y si la probabilidad es mayor de un 10% lanzar una alerta mundial (vale con un System.err.println). Si no, lanzar un mensaje que transmita tranquilidad.

- Mostar por pantalla el tiempo de ejecución total de la aplicación y el tiempo medio de ejecución por cada NEO que se ha analizado.

Para calcular el resultado de la probabilidad de colisión se debe realizar una simulación mediante el siguiente código:

```
double posicionTierra = 1;
double velocidadTierra = 100;
for (int i = 0; i < (50 * 365 * 24 * 60 * 60); i++) {
	posicionNEO = posicionNEO + velocidadNEO * i;
	posicionTierra = posicionTierra + velocidadTierra * i;
}
double resultado = 100 * Math.random() *
	Math.pow( ((posicionNEO-posicionTierra)/(posicionNEO+posicionTierra)), 2);
```

Se pide utilizar una aproximación multiproceso para aprovechar las capacidades hardware de los equipos, ya que existen gran cantidad de NEOs para estudiar y los cálculos son complejos y costosos a nivel computacional. Por tanto, se recomienda paralelizar en la medida de lo posible estos cálculos (es decir, simultanear el cálculo de varios NEOs), teniendo en cuenta para ello el número de cores o procesadores del sistema donde se va a ejecutar la aplicación (el programa debe averiguar el número de cores/procesadores disponibles).

En primera instancia, se pide que la aplicación lea del fichero de datos tantas líneas (cada línea corresponde a la información de un NEO) como cores haya disponibles, los procese en paralelo y muestre los correspondientes mensajes de tranquilidad/alerta.

Como ampliación (se tendrá en cuenta para la nota), se propone que el programa sea capaz de procesar todo el fichero de datos en bloques de líneas (en cada bloque deberá leer tantas líneas como cores haya disponibles), realizar los cálculos, comprobar los resultados y leer el siguiente bloque de líneas. Continuar así mientras haya líneas que leer en el fichero.

Observaciones

- Se entregará en Florida Oberta el fichero ZIP con el código completo de la actividad cuyo nombre seguirá el siguiente formato:
  - <APELLIDO1>_<APELLIDO2>_<NOMBRE>__AE<N>.zip donde “N” es el número de actividad.
- La actividad también deberá subirse a una rama de Git “AE<N>” que partirá de “master”, donde “N” es el número de actividad.

Evaluación

La actividad es obligatoria. Para la evaluación se tendrá en cuenta el funcionamiento de los programas, la codificación adecuada y la documentación. Se puede solicitar al alumno que explique parte de su código así como que realice pequeñas modificaciones.

Recursos

Material de módulo (Florida Oberta).