Escuela Superior de Informática

Computadores Avanzados: Informe N-Cuerpos

Cristian Trapero Mora

06/06/2017

Tabla de contenido

1. Ejecución del código:

Para la compilación y ejecución del código se ha definido un archivo *Makefile* que automatiza dicho proceso. Para compilar ambos códigos existen dos directivas con sus respectivas variantes.

Para compilar el **programa secuencial** ejecutar lo siguiente en una terminal:

- ***make compilar-secuencial***. Compila el programa secuencial con la opción de leer los datos de las variables y los cuerpos desde el fichero datos.dat.
- ***make compilar-secuencial-leyendoParametros.*** Compila el programa secuencial de forma que se lean las variables del programa desde la terminal, y leyendo el valor de los cuerpos desde el fichero datos.dat.

Para compilar el **programa paralelo** ejecutar lo siguiente en una terminal:

- ***make compilar-paralelo***. Esta directiva compila el programa de forma que se impriman los valores de los cuerpos. Los tiempos de ejecución se verán aumentados debido al tiempo dedicado a imprimir la salida de los valores.
- **make compilar-paralelo-sinResultados**. Esta directiva compila el programa sin tener en cuenta en la salida los valores de los cuerpos. Por tanto, no intervendrá en el tiempo de ejecución.

Si se quieren compilar ambos programas a la vez, de forma que lean los parámetros desde fichero y se impriman los resultados, simplemente ejecutaremos en el terminal la instrucción: ***make***

Para lanzar la ejecución de los programas:

- Secuencial: ***make ejecutar-secuencial***
- Paralelo: ***make ejecutar-paralelo***

Para eliminar los ejecutables, ejecutamos: ***make limpiar-directorio***

2. Cuestiones de diseño:

Con lo que respecta a las cuestiones de diseño del **programa secuencial**, he de decir que simplemente he seguido las directrices marcadas en la pp. 14 del enunciado para su desarrollo.

1. Para la lectura de los datos desde fichero he usado dos funciones, una denominada ***leerVariables()*** y ***leerFichero()*.** La primera de ellas lee el valor de las variables desde el fichero, siempre y cuando en las opciones de compilación se especifique, y la segunda lee los valores de los cuerpos.
2. Con lo que respecta al cálculo de la aceleración de los cuerpos, se ha automatizado con una función denominada ***calcularAceleraciones()***, puesto que es llamada dos veces dentro del programa, lo que hace más fácil su legibilidad.

Las directrices de diseño que he seguido para el **programa paralelo** son las siguientes:

1. He creado **dos estructuras** para el manejo de las variables del problema y los datos de los cuerpos. La primera de ellas (*variablesProblema*) contiene todas las variables del problema, que posteriormente se enviaran a todos los procesos con *MPI_Bcast()*. La segunda estructura es la que define una coordenada (*coordenadas*), utilizada para enviar las posiciones, velocidades y aceleraciones de un cuerpo.
2. He definido dos **estructuras *MPI_Datatype*** con las que poder trabajar con las estructuras mencionadas en el punto anterior. Para ello se ha empleado la función *MPI_Type_struct(),* utilizada tanto en *crearEstructuraMPIVariables()* como en *crearEstucturaMPICoordenadas().* El dimensionamiento de dicha estructura usando *MPI_LONG_LONG* está justificado porque al usar *MPI_INT* no dimensionaba correctamente la estructura y a la hora de enviar los datos con MPI_COORDENADAS no seleccionaba la componente y de dicha estructura.
3. Se ha definido la estructura ***MPI_Datatype MPI_CNCR*** que nos permite distribuir los valores de los vectores globales del proceso 0 con la función *MPI_Scatter()* de forma equitativa entre todos los procesos. Para la implementación de dicha estructura se han seguido las directrices del enunciado a partir de la pp. 45-49.
4. Para poder distribuir los datos de los cuerpos entre los distintos procesos, hemos usado la estructura *MPI_CNCR* y la función *MPI_Scatter().* Para que el reparto de dichos datos entre los procesos sea equitativo, ha sido necesario **añadir cuerpos ficticios**, con **masa 0**, de forma que no interactúan con los demás cuerpos. La adición de estos cuerpos se ha implementado en la función ***leerCuerpos(),*** a la cual le pasamos el número de cuerpos que debe de leer, y el número de cuerpos totales que debe crear, añadiendo los ficticios.
5. Para calcular las aceleraciones de los cuerpos se ha desarrollado una función denominada *calcularAceleraciones()* que implementa el algoritmo definido en las pp. 38-39 del enunciado. Dicha función a su vez llama a la función *actualizarAceleraciones()*, la cual es la encargada de actualizar las aceleraciones de a_anillo y a_local.
6. Para que el proceso 0 pueda imprimir los valores de los cuerpos, es necesario que todos los procesos actualicen los datos de los cuerpos en el proceso 0. Para ello se ha utilizado la directiva ***MPI_Gather()*** con la cual actualizamos los vectores globales de los cuerpos del proceso 0, con los vectores locales de cada proceso, para posteriormente imprimirlos.

7. Para la rotación en anillo entre procesos he usado la función *MPI_Sendrecv_replace()* la cual nos evita utilizar buffers intermedios, obteniendo un código más legible y simplificado.