SISTEMAS DISTRIBUIDOS Y PARALELOS 2025 - GRUPO 10

Informe problema de los N-Cuerpos gravitacionales

Garcia Iacovelli, Tobias
Triviño, Gaston

Introducción

En este informe se presentan dos soluciones paralelas para resolver el problema de los N-cuerpos gravitacionales, partiendo del algoritmo secuencial provisto por la cátedra. La primera implementación emplea Pthreads, aprovechando el paralelismo a nivel de hilos en una arquitectura multicore con memoria compartida. La segunda es una solución híbrida que combina MPI y Pthreads, orientada a entornos distribuidos con múltiples nodos, donde cada proceso ejecuta varios hilos concurrentemente.

Ambas estrategias fueron evaluadas en cuanto a su rendimiento y escalabilidad, utilizando métricas como tiempo de ejecución, speedup, eficiencia y overhead, para diferentes tamaños de problema y configuraciones de hardware. A través de un análisis, se busca comparar la efectividad de cada enfoque y determinar bajo qué condiciones se comportan de forma eficiente, considerando tanto el escalado horizontal como vertical del sistema.

Estrategia de paralelización

Para abordar el problema de los N-cuerpos, se partió de una implementación secuencial provista por la cátedra, donde en cada paso de simulación se calcula la interacción gravitacional entre todos los pares de cuerpos. Esta versión fue utilizada como base de referencia para las versiones paralelas.

Paralelización con Pthreads

En la primera solución se utilizaron Pthreads, distribuyendo el trabajo de cálculo de fuerzas y de actualización de posiciones entre múltiples hilos. Para balancear la cantidad de tareas de cada hilo se distribuyeron las mismas usando la estrategia de “stripes”, la cual intercala las tareas una a una entre los diferentes hilos. De esta forma cada hilo es responsable de calcular parcialmente las fuerzas que actúan sobre un subconjunto de cuerpos. La distribución de cuerpos para cada hilo sigue la siguiente lógica:

Tabla 1 - Distribución de cuerpos por hilos.

CANT HILOS	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	
4	H_0	H_1	H_2	H_3	H_0	H_1	H_2	H_3	H_0	H_1	H_2	H_3	H_0	H_1	H_2	H_3	...
8	H_0	H_1	H_2	H_3	H_4	H_5	H_6	H_7	H_0	H_1	H_2	H_3	H_4	H_5	H_6	H_7	

Para calcular la fuerza total de un cuerpo se debe realizar la sumatoria de fuerzas parciales que le son ejercidas por los demás cuerpos, por eso por ejemplo para calcular la fuerza de un cuerpo i se realiza la sumatoria de fuerzas que le ejercen los cuerpos (i+1) hasta N. Por eso elegimos la distribución con stripes, ya que los primeros cuerpos deben realizar el mayor número de cálculos.
Para evitar la condición de carrera que se produciría si un hilo está calculando el par (i,j) y otro el (j,k), se implementó un vector de fuerzas para cada unidad de procesamiento. Estas se suman posteriormente para el cálculo de velocidad y posición.

Paralelización híbrida con MPI + Pthreads

La segunda consiste en una solución híbrida que combina MPI para paralelismo distribuido entre nodos y Pthreads para paralelismo a nivel de hilos dentro de cada nodo. Cada proceso MPI inicializa su conjunto de hilos y trabaja con la mitad de los cuerpos, al saber que se usarán sólo dos máquinas se opto por repartir el calculo de fuerzas intercaladamente entre el nodo A y B. Es decir, el nodo A calcula las fuerzas para los cuerpos pares (0,2,4...) y el nodo B para los impares (1,3,5...). Por otro lado, el calculo de movimiento de cuerpos de divide de manera contigua, el nodo A calcula la primera mitad y el nodo B la segunda mitad de los cuerpos, esto se decidió así para facilitar el envío del vector cuerpos. Antes de comenzar con el cálculo de fuerzas se realiza un broadcast inicial desde el nodo A para compartir la información completa de los cuerpos inicializados.

Durante cada paso de simulación, los procesos sincronizan sus resultados parciales utilizando MPI_Send y MPI_Recv para el intercambio de fuerzas, y MPI_Allgather para actualizar los datos de los cuerpos en todos los nodos.

Análisis de escalabilidad

Para evaluar la eficiencia de las soluciones paralelas se tomaron tiempos de ejecución para distintos tamaños de problema (N = 512, 1024, 2048, 4096) y configuraciones de procesamiento:

- Pthreads con 4 y 8 cores.
- MPI + Pthreads 4 cores: 2 máquinas y 2 hilos en cada máquina.
- MPI + Pthreads 8 cores: 2 máquinas y 4 hilos en cada máquina.
- MPI + Pthreads 16 cores: 2 máquinas y 8 hilos en cada máquina.

Tiempo de ejecución

Tabla 2 - Tiempos de ejecución.

Unidades de procesamiento	Tamaño del problema (N)			
	512	**1024**	**2048**	**4096**
Secuencial	11.3140	45.2903	181.7634	725.7110
Pthread 4	3.4399	13.1418	51.2699	202.0201
Pthread 8	1.9873	6.9095	26.2001	109.5717
MPI+Pth 4	8.6345	31.0716	103.5024	500.7769
MPI+Pth 8	5.7357	16.8703	71.2314	241.0612
MPI+Pth 16	3.1641	9.7738	31.7068	68.9981

Se calculó el speedup y la eficiencia basándonos en el tiempo de la versión secuencial.

Speedup

El speedup S se define como:

$$S = \frac{Tiempo_{Secuencial}}{Tiempo_{Paralelo}}$$

Tabla 3 - Speedups.

Unidades de procesamiento	Tamaño del problema (N)			
	512	**1024**	**2048**	**4096**
Pthread 4	3.29	3.45	3.55	3.59
Pthread 8	5.69	6.55	6.94	6.62
MPI+Pth 4	1.31	1.46	1.76	1.45
MPI+Pth 8	1.97	2.68	2.55	3.01
MPI+Pth 16	3.58	4.63	5.73	10.52

Eficiencia

La eficiencia E se calcula como:

$$E = \frac{S}{p}$$

Siendo p la cantidad de unidades de procesamiento.

Tabla 4 - Eficiencias.

Unidades de procesamiento	Tamaño del problema (N)			
	512	**1024**	**2048**	**4096**
Pthread 4	0.82	0.86	0.89	0.90
Pthread 8	0.71	0.82	0.87	0.83
MPI+Pth 4	0.33	0.36	0.44	0.36
MPI+Pth 8	0.25	0.34	0.32	0.38
MPI+Pth 16	0.22	0.29	0.36	0.66

Análisis para Pthreads

- A partir de la tabla de eficiencia, podemos observar que el programa es fuertemente escalable para tamaños de N iguales a 1024 y 2048 y es casi fuertemente escalable para N=4096.
- También podemos ver que es débilmente escalable al ver cómo se mantiene la eficiencia al aumentar el número de unidades de procesamiento y el tamaño del problema.

Análisis para MPI+Pthreads

- En este caso no es escalable para N chicos, desde 512 a 1024, ya que la eficiencia es baja y no mejora con más hilos.
- Generalmente débilmente escalable, ya que la diagonal de eficiencia se mantiene o incluso aumenta significativamente.
- Para N=4096 es potencialmente fuertemente escalable ya que entre 8 y 16 cores, la eficiencia se duplica.

Conclusiones

A partir de los resultados obtenidos se puede concluir que ambas estrategias de paralelización logran una mejora significativa en el tiempo de ejecución respecto a la versión secuencial.

La implementación basada en Pthreads presentó una muy buena escalabilidad en arquitecturas multicore, con speedup casi lineales y eficiencias superiores al 80% para configuraciones de 4 y 8 hilos. Esto la convierte en una opción eficiente cuando se dispone de una única máquina con múltiples núcleos y memoria compartida.

Por otro lado, la solución híbrida MPI + Pthreads resultó más sensible a la sobrecarga de comunicación entre procesos. El desempeño fue inferior en configuraciones en las que el problema es pequeño, pero escaló considerablemente mejor al aumentar la cantidad de núcleos y el tamaño del problema, mostrando potencial. Debería probarse con tamaños aún mas grandes para tener una mejor conclusión.

El mejor speedup se obtuvo con la configuración híbrida de 16 núcleos y N=4096, lo que confirmaria que el costo de la comunicación se amortiza cuando el volumen de cómputo por nodo es suficientemente alto.

En definitiva, la elección de la estrategia de paralelización más adecuada depende de la arquitectura de hardware disponible y del tamaño del problema. Para simulaciones medianas en una sola máquina, en este caso, Pthreads es la opción más eficiente. Mientras que para ejecuciones de problemas grandes con bastantes unidades de procesamiento, la solución híbrida MPI+Pthreads permite explotar de mejor forma los recursos.