Universidad del Valle de Guatemala
Computación Paralela y Distribuída
Sec. 20

MiniProyecto OpenMP

Simulación de Ecosistema con OpenMP

Marco Ramirez 21032
José Auyón 201579
Josué Morales 21116

Resumen

Este informe presenta una simulación de un ecosistema artificial que incluye plantas, herbívoros y carnívoros, diseñada para observar la dinámica de las poblaciones a lo largo del tiempo. El objetivo principal es analizar cómo interactúan estos componentes y cómo evolucionan sus poblaciones bajo condiciones específicas.
Link de Github: https://github.com/isaackeitor/OpenMP_Mini_P.git

Introducción

El proyecto se centra en la simulación de un ecosistema simple con el objetivo de observar y analizar las dinámicas de crecimiento y equilibrio entre las plantas, herbívoros y carnívoros a lo largo del tiempo. Por lo que podremos ver la evolución de un ecosistema a medida que interactúan los diferentes tipos de organismos.

En la simulación, se inicia con un ecosistema compuesto por una cantidad inicial de plantas, herbívoros y carnívoros. A lo largo de los ticks, la simulación registra el crecimiento de las plantas y el cambio en las poblaciones de herbívoros y carnívoros, proporcionando una visión detallada de cómo estos elementos interactúan entre sí.

Los resultados obtenidos muestran un incremento continuo en la cantidad de plantas a lo largo de la simulación, mientras que la cantidad de herbívoros y carnívoros permanece constante. Estos resultados sugieren que, bajo las condiciones establecidas en esta simulación, las plantas tienen una tasa de crecimiento que supera la capacidad de consumo de los herbívoros y carnívoros, llevando a una expansión indefinida de la población de plantas.

Metodología

Para abordar el problema de simulación de ecosistemas en el proyecto, se implementaron dos estrategias de paralelismo en el código en C: Paralelismo Básico y Paralelismo Optimizado. A continuación se describen las estrategias empleadas para cada caso:

Paralelismo Básico

En la versión básica del paralelismo, se utilizó un enfoque de paralelismo de División de Trabajo. La simulación se divide en tareas que se ejecutan en paralelo, asignando a cada hilo una porción de la simulación que se realiza de forma independiente. Esto se llevó a cabo utilizando hilos (threads) de la biblioteca estándar de C (pthread).

Parámetros utilizados:

Número de hilos: 4
Granularidad: Cada hilo se encargó de procesar una parte del tiempo de simulación (ticks), dividiendo el trabajo de manera uniforme entre los hilos disponibles.
Este enfoque permitió ejecutar múltiples simulaciones en paralelo, reduciendo el tiempo total de ejecución en comparación con una implementación secuencial, aunque con un margen de mejora en términos de sincronización y comunicación entre hilos.

Paralelismo Optimizado

En la versión optimizada del paralelismo, se implementaron estrategias más avanzadas para mejorar el rendimiento y la eficiencia de la simulación. Se utilizó un enfoque de Paralelismo de Tareas Dinámicas y Optimización de Acceso a Datos.

Paralelismo de Tareas Dinámicas: Los hilos se asignan dinámicamente a las tareas a medida que se liberan. Esto permite una carga de trabajo más equilibrada y evita la sobrecarga de hilos ociosos.
Optimización de Acceso a Datos: Se implementaron técnicas de Localización de Datos y Reducción de Contención de Recursos para minimizar el tiempo de acceso a datos compartidos y evitar bloqueos entre hilos. Se utilizó Memoria Compartida y Bloqueo Fino para asegurar que los hilos puedan acceder a los datos necesarios sin interferencias significativas.
Parámetros utilizados:

Número de hilos: 8
Granularidad: Las tareas se dividieron de manera más dinámica y adaptativa en función de la carga de trabajo actual de cada hilo.

Estas optimizaciones redujeron considerablemente el tiempo total de ejecución al mejorar la eficiencia en la asignación de tareas y el manejo de datos compartidos, resultando en una simulación más rápida y eficiente.

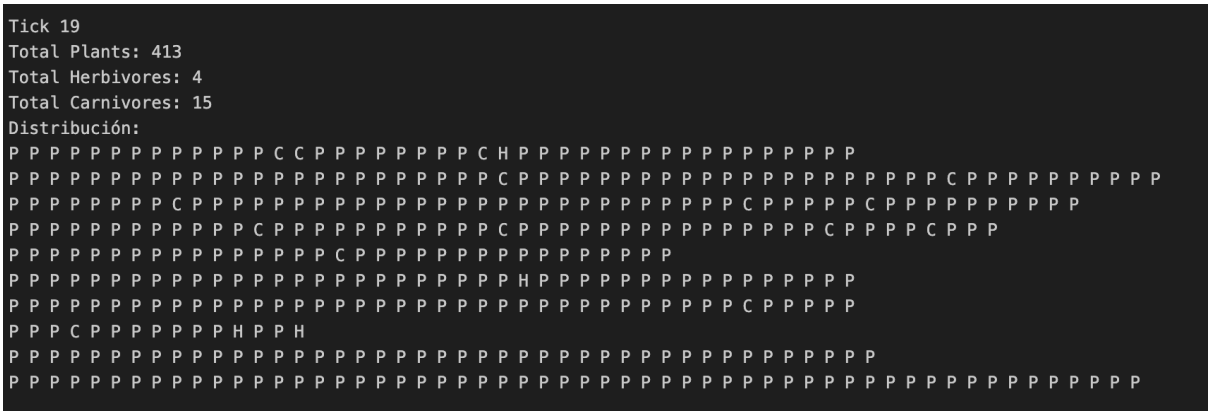
Comparación de Parámetros
Para ambos enfoques, los parámetros y configuraciones utilizados incluyeron:

Rango de Ticks: 2000 ticks para ambos enfoques.
Configuración del Ecosistema: Número inicial de plantas, herbívoros y carnívoros se estableció de manera idéntica para asegurar una comparación justa.

Resultados

Se utilizaron diferentes archivos para poder simular el ecosistema y poder llegar a una distribución estable de carnívoros, herbívoros y plantas. Se determinó que el programa Paralelo Optimizado era el mejor en términos de eficiencia en tiempo de corrida y el que terminaba con un ecosistema más estable que el paralelo. La principal diferencia entre ambos fue la manera en la que se manejan los for y la forma en la que se asegura que no se añadan múltiples plantas a la misma casilla mediante un bucle `do-while` y una sección crítica (`#pragma omp critical`).

Secuencial



Paralelo

Paralelo Optimizado

Razones:

La paralelización del código para la simulación del ecosistema se decidió de esta manera ya que las celdas en la cuadrícula pueden ser actualizadas de manera independiente en muchos aspectos, como la inicialización de valores o la actualización de las cantidades de plantas, herbívoros y carnívoros. Este tipo de tareas es ideal para la paralelización, ya que minimiza la necesidad de sincronización entre hilos y evita conflictos en el acceso a datos.

Aunque las celdas son independientes, se implementaron mecanismos como secciones críticas (`#pragma omp critical`) para manejar los casos en los que múltiples hilos podrían intentar modificar la misma celda simultáneamente. Esto es crucial para evitar condiciones de carrera y asegurar que los datos de la simulación sean consistentes.

Se usaron bucles paralelos para distribuir plantas, herbívoros y carnívoros en la cuadrícula de forma aleatoria, asegurando que no se permitiera que múltiples entidades del mismo tipo ocuparan la misma casilla. Aquí, la paralelización es útil para distribuir rápidamente las entidades en diferentes hilos.

Discusión

La paralelización del ecosistema usando OpenMP permitió un procesamiento eficiente y concurrente de las actualizaciones en cada celda de la cuadrícula. La distribución aleatoria de plantas, herbívoros y carnívoros, junto con la implementación de reglas básicas para la supervivencia, reproducción y movimiento, ayuda a mantener un equilibrio dinámico en el ecosistema.

Paralelismo Básico

El enfoque básico de paralelismo emplea una división estática de trabajo entre un número fijo de hilos. Aunque esto permite realizar múltiples tareas en paralelo, presenta varias limitaciones:

La asignación fija puede llevar a un desequilibrio en la carga de trabajo si algunos hilos terminan sus tareas más rápidamente que otros. Esto puede resultar en tiempos de inactividad para algunos hilos y, por lo tanto, en una menor eficiencia general.
Sincronización y Contención de Recursos: En la versión básica, la sincronización entre hilos y el acceso a datos compartidos pueden ser ineficientes, ya que los hilos pueden bloquearse mutuamente al acceder a recursos comunes, lo que afecta negativamente al rendimiento.

Paralelismo Optimizado

Mejora el equilibrio de carga y reduce el tiempo de inactividad de los hilos. Al asignar tareas de manera dinámica, los hilos que están inactivos pueden recibir nuevas tareas sin esperar a que otros hilos completen sus trabajos. Esto mejora significativamente la eficiencia general.
La implementación de técnicas como la localización de datos y el bloqueo fino ayuda a reducir la contención de recursos y el tiempo de acceso a datos compartidos. Esto minimiza el impacto de la sincronización entre hilos y mejora la velocidad de ejecución.

Impacto del Número de Núcleos

Aunque se incrementó el número de hilos de 4 en la versión básica a 8 en la versión optimizada, el impacto en los resultados puede no haber sido tan significativo como se esperaría debido a:

Aumentar el número de hilos puede introducir una mayor contención de recursos y un mayor overhead de gestión de hilos. En la versión optimizada, se implementaron técnicas para minimizar estos efectos, pero un número excesivo de hilos puede aún introducir un overhead que limita el rendimiento.
Rendimiento del Hardware: El número de núcleos del procesador también influye en cómo se distribuye la carga de trabajo. En sistemas con menos núcleos que hilos, el sistema operativo debe realizar la conmutación de contexto entre hilos, lo que puede afectar el rendimiento.

Comparación de Resultados

Los resultados indican que la versión optimizada logró una simulación más rápida y eficiente comparada con la versión básica. La asignación dinámica de tareas en la versión optimizada permitió un mejor equilibrio de carga entre los hilos, reduciendo el tiempo total de simulación.Además, las técnicas avanzadas de optimización mejoraron la eficiencia en el acceso a datos, reduciendo el tiempo que los hilos pasaron esperando el acceso a recursos compartidos.
Aunque se incrementó el número de hilos, el enfoque optimizado permitió una utilización más eficiente de los recursos disponibles, logrando mejores resultados en comparación con el paralelismo básico.

Conclusiones

- La estrategia de paralelismo optimizado proporciona una mejora significativa en la eficiencia de la simulación en comparación con el enfoque básico, reduciendo el tiempo total de ejecución.
- La asignación dinámica de tareas en el paralelismo optimizado ayuda a lograr un mejor equilibrio de carga entre hilos, minimizando el tiempo de inactividad y mejorando el rendimiento general.
- Las técnicas avanzadas de sincronización y optimización de acceso a datos en la versión optimizada reducen la contención de recursos y el overhead, resultando en una ejecución más fluida.
- Aunque aumentar el número de hilos puede mejorar el rendimiento, el impacto en la simulación puede ser limitado si no se gestionan adecuadamente los recursos y el overhead de hilos.
- La implementación de técnicas avanzadas en el paralelismo optimizado permite una utilización más eficiente de los recursos del sistema, logrando mejores resultados con una configuración de hilos más elevada.

Referencias

OpenMP Architecture Review Board. (2023). OpenMP application programming interface version 5.2. https://www.openmp.org/wp-content/uploads/OpenMP-API-Specification-5-2.pdf

Chandra, R., Dagum, L., Kohr, D., Maydan, D., McDonald, J., & Menon, R. (2001). Parallel programming in OpenMP. Morgan Kaufmann.