Universidad del Valle de Guatemala

Realizado por: Brandon Reyes Morales 22992

Mini mini proyecto trafico

Link repositorio: https://github.com/BrandonReyes0609/mp-trafico-paralela.git

Descripción del problema

El proyecto consiste en simular el comportamiento de trafico vehicular regulad por semáforos. Cada vehículo avanza dependiendo del esta del semáforo que le corresponde. Se realizaron dos versiones, una secuencial y otra paralela:

Estrategia de paralelización utilizada:

Para la versión paralelizada OpenMP con la directiva "#pragma omp parallel for" para distribuir el procesamiento de movimiento de los vehículos entre múltiples hilos. La idea fue asignar a cada hilo una porción de vehículos, de forma dinámica usando la estrategia "Schedule(dynamic)", lo cual permite balancear la carga de manera eficiente y aprovechar mejor los núcleos disponibles.

Además, se considero el uso de paralelismo dinámico "omp_set_dynmaic(1)" para permitir que el número de hilos sea ajustado automáticamente la ejecución según la carga. Tambien se implementaron versiones con "omp_set_nested(1)" para evaluar el comportamiento con paralelismo anidado, aunque no fue la estrategia principal seleccionada.

Justificación con el uso de OpenMP

OpenMP fue utilizado porque proporciona una forma sencilla y podesora de paralelizar código en C. Permite aprovechar múltiples núcleos del procesador sin necesidad de gestionar explícitamente los hilos. En este proyecto, se empleo el paralelismo dinámico con "Schedule(dynamic)" para mejorar el reparto de trabajo cuando la cantidad de vehículos no se divide equitativamente entre los hilso.

El paralelismo anidado se activo se activo con "omp_set_nested(1)" para evaluar estructuras jerárquicas de trabajo, aunque en esta simulación no se encontraron grandes beneficios con esta estrategia. En general, el uso de OpenMP permitió un programa secuencial en uno eficiente y escalable, mantenido la lógica original de forma sencilla.

Simulación Secuencial

- En cada iteración, se evalúa el estado del semáforo y se actualiza la posición del vehículo uno por uno.
- No hay indicación de qué vehículos fueron evaluados o si fueron bloqueados por el semáforo (esto es una desventaja en términos de visibilidad de comportamiento).
- La mayoría de vehículos llegan como máximo a la posición 4, lo cual indica que el flujo de tráfico fue controlado por los semáforos, y no hubo condiciones artificiales que impidieran su movimiento.

Simulación Paralela

- Se utilizó #pragma omp parallel for schedule(dynamic) para ejecutar la lógica de movimiento de vehículos con múltiples hilos.
- Se puede observar explícitamente qué hilo movió (o no) a qué vehículo. Esto permite detectar:
    - Qué vehículos están bloqueados por el semáforo en ROJO o ÁMBAR.
    - Qué vehículos sí lograron avanzar gracias a un semáforo VERDE.
- Al final de las 10 iteraciones, los vehículos también alcanzan posiciones máximas similares a la secuencial (posición 4 o menos).
- Se logró una simulación más detallada y observación concurrente del comportamiento.

Comparación de Rendimiento

- La versión paralela está mejor equipada para escalar a un número mayor de vehículos, ya que reparte la carga de trabajo entre hilos.
- La ejecución paralela mejora el rendimiento y la observabilidad, pero solo si se evita el acceso concurrente conflictivo (en este caso no hubo colisiones porque cada hilo trabaja con un vehículo distinto).

Conclusiones

- Exactitud del modelo: Ambas simulaciones reflejan el mismo comportamiento lógico, los vehículos solo avanzan si el semáforo correspondiente está en VERDE.
- Paralelización efectiva: El uso de #pragma omp parallel for schedule(dynamic) en la versión paralela permite repartir dinámicamente las tareas entre múltiples hilos, aumentando la eficiencia sin alterar la lógica del modelo.