Instituto Tecnológico y de Estudios Superiores de Monterrey

Campus Querétaro

Actividad Integradora 5.3 Resaltador de sintaxis paralelo (evidencia de competencia)

Implementación de métodos computacionales (Gpo 601)

Santiago de Querétaro, Querétaro, 16 de marzo del 2022

Dr. Pedro Oscar Pérez Murueta

Aldo Tena García
Renato Sebastián Ramírez Calva

A01275222
A01275401

A01275222@itesm.mx
A01275401@itesm.mx

Manual del usuario

Antes de compilar este proyecto es necesaria asegurarse de que los archivos: “Comentario.hpp”, “Entero.hpp”, “Reales.hpp”, “Variable.hpp”, “String.hpp”, “PalabraReservada.hpp”, “utils.h”, “input1.txt”, “input2.txt” y “styles.css”. Todos los archivos anteriores serán necesarios para la correcta compilación de main.cpp, es importante aclarar que input1.txt y input2.txt serán usados como archivo de ingreso para ser analizados por el resaltador de sintaxis por lo que puede ser reemplazados.

Para compilar este programa es necesario ubicarse dentro de la carpeta donde estén todos los archivos anteriormente mencionados e ingresar en la terminal: g++ .\main.cpp -lpthread -o app, siendo app el nombre recomendado para el archivo .exe, este puede ser cambiado. Para ejecutar este programa es necesario ingresar en la terminal: \app input1.txt input2.txt, en donde los input pueden ser reemplazados por el archivo que se quiere ingresar al resaltador de sintaxis, por el momento este programa solo puede soportar 2 inputs a la vez, estos serán procesados por 2 hilos de trabajo.

Los resultados del programa serán un total de 3 archivos HTML, 1 por parte del resaltador de sintaxis en paralelo y 2 por parte del resaltador en paralelo, 1 por cada hilo de trabajo. El HTML de la versión secuencial y el primer HTML de la versión paralela serán iguales.

Cálculo de la complejidad

La complejidad general de este algoritmo se tiene que calcular tomando en cuenta el peor de los casos de todas las funciones que fueron implementadas y el número de iteraciones que tuvo el programa:

- Lectura del archivo: O(n)
- Asignación de clases: O(1)
- Escritura del archivo: O($n^5 log(n)^2$)
- Asignación y uso de los hilos de trabajo: O(n)

Una vez que se tomaron en cuenta todos las funciones anteriores, claro basándonos en el peor caso de nuestro programa, tomamos en cuenta no sólo los ciclos anidados si no también las excepciones a los casos (else y elseif), con lo que se puede llegar a la conclusión de que la complejidad del algoritmo que hemos realizado sería de: O($n^5 log(n)^2$), la escritura del archivo es la parte más compleja no solo tenemos que repetir la iteraciones cierta cantidad de veces bajo condiciones sino que tenemos que ver bajo las excepciones que cada línea de nuestra entrada puede tomar, realizar un lexer implica un minucioso análisis no solo parte de la computadora sino del programador ya que se deben tener en cuenta cada posible caso que se puede presentar por lo que la complejidad del algoritmo se puede aseverar.

Reflexión sobre las soluciones planteadas

A lo largo del desarrollo de esta actividad pudimos darnos cuenta de los beneficios que supone la programación paralela haciendo uso de los múltiples núcleos que están presentes en los CPU´s contemporáneos. El poder procesar múltiples archivos o tareas de manera simultánea mejora en un gran medida la eficiencia del programa desarrollado así como su posibilidad de realizar tareas cada vez más complejas.

Con la complejidad obtenida(que está basado en el análisis de un solo archivo) la compilación de manera concurrente presenta mucha carga para un solo hilo de trabajo, por lo que la compilación en paralelo la cual distribuye un archivo para cada hilo declarado permite a la optimización y velocidad del equipo.

Es importante tener en consideración que actualmente se trabaja con una gran cantidad de tareas e información en prácticamente todos los programas y sistemas de la actualidad y poder realizar este tipo de implementaciones resulta de suma importancia para poder hacer uso de todos los recursos disponibles para resolver el problema de la manera más eficiente posible.

Implicaciones éticas

Es importante tener en cuenta que todas las tecnologías que son desarrolladas tienen implicaciones éticas la cuales repercuten en la sociedad aunque uno no lo considere, a pesar de que este tipo de programas y de herramientas se generan pensando en la optimización de tareas no siempre son buenas dichas tareas ya que podrían derivar, por ejemplo, en programas de varios hilos para posibles ataques cibernéticos o para ataques de fuerza bruta a contraseñas. En primera instancia podría suceder que debido a una mala gestión de los hilos se generen daños físicos al procesador de la computadora y se podría escalar este tipo de situaciones, pero también es importante tomar en cuenta todos los beneficios que se tienen del trabajo de múltiples hilos de trabajo en la ejecución de un código y de como esta hace posible que programas cada vez más complejos pueden ejecutarse en menos tiempo, como todos los cálculos y tareas involucrados en la renderización de un videojuego.

A final de cuentas puede verse todo lo anteriormente mencionado como una herramienta con la que interactúa el usuario y éste mismo quien decide qué enfoque y qué implicaciones tiene aquello que genero.