Junio 2022

Actividad Integradora 5.3 Resaltador de sintaxis paralelo

Instituto Tecnológico y de Estudios Superiores de Monterrey
Campus Querétaro

Implementación de métodos computacionales

TC2037.601

Pedro Oscar Pérez Murueta

Presenta:

Said Ortigoza Trujillo | A01707430
Miguel Weiping Tang Feng | A01610836

Enlace al repositorio de Github

En el mismo archivo Integradora5.3.cpp se encuentra la complejidad del código
https://github.com/saidortigoza/Integradora5.3_TC2037

Ejecución de pruebas

Caso de prueba 1:
Utilizando 22 archivos y 8 hilos obtenemos los siguientes resultados:

```
Ejecutando forma secuencial...
Ejecutando forma concurrente...
------------Secuencial--------------
Tiempo = 90562
------------Multihilo--------------
Tiempo Promedio = 4007.9
```

Secuencial = 90562 ms
Concurrente ~ 4007.9 ms

Speed up:

$$S_8 = \frac{T_1}{T_8} | S_8 = \frac{90562}{4007.9} | S_8 = 22.59$$

Esto quiere decir que para 22 archivos utilizando 8 hilos utilizando la programación concurrente obtenemos que es 22 veces más rápido que la secuencial

Caso de prueba 2:
Utilizando 9 archivos y 8 hilos tenemos los siguiente resultados:

```
Ejecutando forma concurrente...
------------Secuencial--------------
Tiempo = 32559
------------Multihilo--------------
Tiempo Promedio = 2735.6
```

Secuencial = 32559 ms
Concurrente ~ 2735.6 ms

Speed up:

$$S_8 = \frac{T_1}{T_8} | S_8 = \frac{32559}{2735.6} | S_8 = 11.901$$

Lo anterior implica que utilizando la programación concurrente con 8 hilos para 9 archivos es casi 12 veces más rápida que la secuencial.

Caso de prueba 3:
Utilizando solo un archivo con 8 hilos de trabajo, se tienen los siguientes resultados

```
Ejecutando forma secuencial...
Ejecutando forma concurrente...
------------Secuencial--------------
Tiempo = 1911
------------Multihilo--------------
Tiempo Promedio = 1952
```

Secuencial = 1911 ms
Concurrente ~ 1952 ms

Speed up:

$S_8 = \frac{T_1}{T_8} | S_8 = \frac{1911}{1952} | S_8 = 0.02561 \ \ | \ S_1 = \frac{T_8}{T_1} | S_1 = \frac{1952}{1911} | S_1 = 1.021$

Esto significa no hay un speedup cuando se utiliza un solo archivo, de hecho es más rápido utilizar la manera secuencial teniendo un speed up de 1.021

Conclusión y reflexiones

Con base en los resultados obtenidos en la ejecución de pruebas, podemos concluir que podemos aumentar la eficiencia de nuestro resaltador de sintaxis utilizando multihilos, ya que la ejecución concurrente de hilos de un mismo proceso puede mejorar la eficiencia del sistema, debido a que pueden aprovechar mejor la capacidad completa de los procesadores modernos, por consiguiente, reducir el tiempo de ejecución de manera considerable.

Algo que observamos es que la eficiencia de la ejecución secuencial sobre la paralela y viceversa depende del número de tareas que el código tenga que ejecutar, mientras más archivos sean incluidos en la operación, la forma concurrente resulta mucho más eficiente, mientras que cuando reducimos el número de archivos, el tiempo de ejecución utilizado por el algoritmo multihilo es ligeramente mayor al secuencial. Finalmente, cuando se trabaja con un archivo, el tiempo de ejecución es similar.

Algunas implicaciones éticas que podría tener nuestro programa, es en el aspecto de qué tan accesible resulta, es probable que una persona con capacidades visuales diferentes, como el daltonismo, que en la mayoría de los casos se debe a problemas genéticos, tenga problemas para diferenciar los colores asignados a cada tipo de expresión, ya que utilizamos colores azules, verdes, rojos y naranjas principalmente, y la afección provoca que solamente puedan ver sombras de gris, ocasionando que no puedan diferenciar de manera sencilla los diferentes tipos de expresiones que incluimos, por lo que sería una implicación ética que nuestra tecnología podría tener.