Universidad de Costa Rica

Escuela de Ciencias de la Computación e Informática

Programación Paralela y Concurrente – CI0117

Examen 1

Profesor Jose Andrés Mena Arias

Estudiante Gianfranco Bagnarello Hernández

Responda las siguientes preguntas (20 pts)

1. **Describa brevemente cuál fue la sección (o secciones) del programa que resolvió utilizando programación paralela, indicando el o los métodos de sincronización utilizados y la zona crítica.**

Para calcular las ocurrencias en las secuencias de datos, se crearon dos métodos, uno para cada una, ya que se requería distinción entre las estructuras que cada uno estaba modificando. Estas funciones (*get_first_sequence_occurrences* y *get_second_sequence_occurrences)* son las llamadas por los hilos creados. Dentro de ambas funciones, existe una línea donde se escribe información dentro de una estructura que todos los hilos comparten y pueden acceder (*shared_data*). Esta es la zona crítica que se escogió proteger, en este caso con un mutex que se bloquea antes de la escritura, y una vez el hilo que lo bloqueó termina de escribir, se desbloquea, para que el próximo pueda.

2. **Con respecto a los enfoques vistos en clase: optimización y separación de asuntos, ¿Cuál es el enfoque en el que se centra este problema? Explique.**

Este problema se clasifica como uno de optimización, ya que se está intentando manejar un enorme volumen de datos, en este caso puede llegar a ser desde 1 hasta 10^8 caracteres en la secuencia, por lo que la concurrencia por paralelismo puede ayudar bastante con el desempeño. El enfoque del problema es

principalmente en distribuir el trabajo de la manera más equitativa posible entre la cantidad de hilos indicados por el usuario.

3. **¿Considera que la solución paralela es siempre la mejor para resolver este tipo de problemas? Explique.**

Dependiendo del problema, puede que no sea necesario implementar una solución de este tipo como el caso en el que el rendimiento serial y el concurrente sean muy parecidos. Existen tambien otros casos donde el problema puede ser bastante simple y no requiere de optimización ni de división de trabajo. Un ejemplo puede ser cuando se trabaja con pequeñas cantidades de datos, o pocas iteraciones; dichos problemas se pueden implementar con hilos, pero la diferencia en rendimiento va a ser casi nula y puede complicar bastante la solución.

Problema 2 [cigarrete_smoker]

Describa cuál fue el cambio realizado. Explique brevemente si considera que el cambio realizado es mejor o peor en algún aspecto. Si considera que no hay diferencia, explique por qué (10 pts).

Se realizó un cambio del semáforo binario "agente" a un mutex, que trabaja en conjunto con una variable de condición cond_var, la cual espera a que un valor booleano "fumando" sea verdadero, para despertar al hilo entrante y que este ejecute su función. En caso contrario pone el hilo a dormir. Considero que el cambio a un mutex funciona igual que el semáforo, ya que en función un semáforo binario es lo mismo que un mutex solo que con diferente sintaxis. Añadirle una variable de condición puede hacer que el programa sea más seguro ya que verifica que el booleano sea verdadero, pero creo que si solo se utilizara el mutex sin la variable, funcionaría mejor.