INFORME

ENTREGA 1

Optimización de algoritmos secuenciales

Nehuen Pereyra 878/6

Soledad Rocha 773/7

Grupo 9

16 de Abril de 2021

Sistemas Distribuidos y Paralelos

ENUNCIADO

Resolver la expresión de álgebra de matrices de tamaño NxN. Se cuenta con la siguiente expresión a hacer calculada:

$$\mathbf{R = (AU + LB + CA)/5 + Fib(F)}$$

Donde:

- L y U son matrices triangulares de NxN con valores aleatorios de tipo doble.
- A, B y C son matrices de NxN con valores aleatorios de tipo doble.
- F es una matriz con características especiales que debe ser inicializada
- con elementos de algún tipo entero. Esta con valores aleatorios en un rango de 1 a 40.
- Fib(F) es una función que aplica Fibonacci a cada elemento de la matriz F.

Se solicita obtener los tiempos de ejecución de calcular la expresión almacenando los ceros en las matrices triangulares y sin almacenarlos. Para validar la ejecución deberán compararse los resultados de las dos soluciones en el mismo archivo. Por último, evaluar la expresión para los siguientes tamaños de matrices potencias de 2: N=512, N=1024 y N=2048.

RESOLUCIÓN

Para la resolución del enunciado, se utilizaron los siguientes códigos fuentes en lenguaje c provistos en la práctica 1 de la cátedra:

- **matrices:** utilizamos el concepto de almacenar por filas o por columnas las matrices. Para la multiplicación se recorre la primera matriz por filas mientras la segunda por columnas de este modo se logra reducir los fallos de memoria caché. Además se incorpora una variable de tipo register (se le indica al compilador una preferencia para que la variable se almacene en un registro de la CPU, si es posible, con el fin de optimizar su acceso) que permite ir almacenando las sumas parciales de la multiplicación de esta manera se reducen los retardos de ir a

memoria al guardar la matriz.

- **expMatrices1**: utilizamos el concepto de dividir el problema de multiplicar matrices y sumarlas en resolver cada multiplicación con un una instrucción for y luego sumar las matrices intermedias utilizando una última instrucción for. Esto es más eficiente que realizar toda la operación en un solo bucle.
- **triangular**: utilizamos el concepto de definir el tamaño de una matriz triangular sin almacenar los ceros. Además de almacenar una matriz triangular inferior por filas como también una matriz superior por columnas. Para realizar la multiplicación definimos el rango de k.
- **instrucciones2**: utilizamos el concepto de que es más eficiente realizar una multiplicación que una división, así que cuando se pueda se debe optar por realizar la operación de multiplicar.
- **fib**: utilizamos el concepto de que es más eficiente realizar la operación de fibonacci de forma iterativa ya que si se realiza de manera recursiva se realizan llamados a la función, la cual debe ir apilando los parámetros a la pila y eso se traslada en generar un retraso.
- **iterstruct**: utilizamos el concepto de que para realizar la multiplicación de matrices es más eficiente realizarla con un bucle for que con un while ya que con el uso de 'for' el compilador puede analizar el código y evitar los saltos, cosa que con el while no le es posible.

Como se puede observar en el archivo del código, fueron creadas diversas matrices para inicializar las matrices principales y luego unas secundarias para almacenar las primeras multiplicaciones AU, LB y CA.

Se puede describir al código en tres bloques:

- Cálculo de las multiplicaciones de las matrices AU, LB y CA.
- Se suman las matrices intermedias AU+LB+CA y luego se divide por 5 cada elemento (esto se realiza multiplicando por 0.2 ya que es más eficiente).
- Se aplica para cada elemento de la matriz F la operación de Fibonacci y a la matriz resultante se le suma la obtenida en la etapa anterior.

Para la resolución final, se hizo dos cálculos

a) Solución de matrices triangulares almacenando los ceros
b) Solución de matrices triangulares sin almacenar los ceros

Para el caso a), el tamaño de las matrices son de N*N. Para el segundo caso, el tamaño de las matrices es de N * (N+1)/2. Con este método, el objetivo es minimizar el espacio de almacenamiento evitando guardar los elementos de la parte nula de las matrices. Por lo tanto, no accederemos a las posiciones nulas de estas.

A continuación, los siguientes resultados se encuentran en segundos. En los tres casos, las operaciones se resolvieron correctamente.

Tabla 1. Resultados de la multiplicación de matrices.

N	Tiempo total CON ceros	Tiempo total SIN ceros	Diferencia
512	2.018974	2.084773	0.065799
1024	16.326912	16.806574	0.479662
2048	129.945059	134.175132	4.230073

Como puede observarse en la tabla, a medida que N se incrementa la diferencia aumenta también. Por lo que, el tiempo que posee el almacenar y recorrer las posiciones donde se encuentran los ceros es mucho mejor que no recorrerlos. Esto puede ser causado debido al cálculo adicional que se realiza para obtener las posiciones que se necesitan.

En resumen, a partir de lo analizado se ha optado por seleccionar el algoritmo que incluye ceros como mejor algoritmo secuencial dados los tiempos obtenidos.