Universidad Del Valle | Sede Tuluá

Fundamentos de programación funcional y concurrente

Ingeniería en Sistemas

Taller 4 Programación

Carlos Andrés Delgado Saavedra

Alejandro Sierra Betancourt 2259559

Juan Pablo Castaño Arango 2259487

Juan Manuel Ramírez Agudelo 2259482

5 diciembre 2023

Informe corrección de las funciones:

1.1 Multiplicación estándar de matrices:

- **matrizAlAzar:** - Se reciben dos valores de tipo entero, el primero es la longitud de la matriz y el segundo los valores que toma y retorna una matriz.
  La función realiza un relleno aleatorio de valores a partir de los dos parámetros anteriores.

- **vectorAlAzar:** - Se reciben dos valores de tipo entero, el primero siendo la longitud del vector y el segundo los valores que toma el vector, retornando un vector.
  La función genera un vector aleatorio con los parámetros mencionados anteriormente.

- **prodPunto**: - Recibe dos vectores como parámetros y retorna un entero.
  Esta función usa las funciones zip, map, y sum para hallar el producto punto entre dos vectores.

- **prodPuntoPar:** -Para realizar esta función se implementó una función que nos permite usarla como paralela, siendo esta "future**"** la cual aplica cálculos asincrónicos.

- **traspuesta: -** La función transpuesta recibe una matriz y devuelve una matriz.
  La función que cumple es intercalar las filas por las columnas y viceversa.

1.1.1 Versión estándar secuencial:

- **multMatriz: -** Recibe dos matrices y retorna una matriz resultante del mismo tamaño que vendría siendo la multiplicación entre ellas, usando las funciones anteriormente mencionadas teniendo en cuenta la explicación de como resolverlo en el enunciado del taller.

1.1.2 Versión estándar paralela:

- **multMatrizPar: -** Se aplica la misma lógica que la versión multMatriz, solo que en este caso se hace uso de la función paralela "task" para representar una tarea que se realiza en paralelo.

1.2 Multiplicación recursiva de matrices secuencial:

1.2.1 Versión recursiva secuencial:

-Se establecen en primera instancia las dimensiones de la matriz, una vez hecho esto se inicializa la matriz resultante.
La sección "Vector.tabulate" es responsable de generar cada columna j en la matriz de resultado. Dentro de este bloque, se lleva a cabo el cálculo de la multiplicación de matrices mediante el segmento "(0 until colsA).map(k => a(i)(k) * b(k) (j)).sum", este segmento realiza una serie de operaciones detalladas a continuación.

-0 until colsA: -Representa la iteración a través de los índices de las columnas en la matriz a, ya que colsA denota el número de columnas en a.

-a(i)(k)*b(k)(j): -Multiplica los elementos correspondientes de la fila i de la matriz a y la columna j de la matriz b.

-.sum: -Calcula la suma de estos productos, lo que resulta en la obtención del valor de un elemento específico ubicado en la posición (i,j) de la matriz resultante.

1.2.2 Versión recursiva paralela:

- Primero, se inicializa un conjunto de futuros, futures, que representa cada fila de la matriz resultante. Dentro de cada futuro se ejecuta un cálculo con "Vector.tabulate(colsB) {j => ... }", el cual multiplica y suma los elementos correspondientes de las matrices de entrada.

Luego, se combina este conjunto de futuros en un único futuro, resultFuture, mediante Future.sequence(futures), lo que permite esperar la finalización de todos los cálculos en paralelo.

Por último, "Await.result(resultFuture, 5.seconds)" se encarga de esperar y obtener el resultado final después de que todos los cálculos paralelos se completen, con un límite de tiempo de 5 segundos.

1.3 Multiplicación de matrices usando el algoritmo de Strassen:

-La lógica empleada para esta función es la misma para la versión estándar como para la versión paralela.

La lógica que se lleva es dividir las dos matrices para hacer la operación, en este caso serían sumas y restas, una vez terminada la suma y la resta, se multiplican entre ellas para al final sumarlas y que nos de la matriz resultante.

Todo esto se hace mediante las funciones "subMatriz", "restarMatriz" y "sumMatriz". La función como tal es una función recursiva que dependiendo de sus parámetros se van generando nuevas submatrices. Después de esto se tienen en cuenta las condiciones de las filas y columnas, para agruparse entre ellas.

- **subMatriz:** -Lo que hace esta función es generar una matriz que pertenece a una matriz original.
- **restarMatriz y sumMatriz:** -Se tiene en cuenta la posición de la primera y la segunda matriz, generando la matriz resultante.

Para comprobar que los datos coincidan en todas las versiones para multiplicar matrices, se realizaron diferentes test donde se demuestra que su resultado siempre es el mismo, aplicando diferentes tamaños y diferentes versiones siendo tanto estándar como paralela de los tres tipos.

Todos estos test fueron implementados gracias al gradel que se encuentra dentro del repositorio.

Desempeño de las funciones secuenciales -paralelas

Se realizaron las pruebas para calcular el tiempo de ejecución de cada una de las versiones de multiplicar matrices, la implementación se hace mediante un for que altera los parámetros de la función matrizAlAzar y por cada iteración del bucle va mostrando un resultado. Se debe tener en cuenta que el tamaño de la matriz es de 2^n

multiMatriz y multiMatrizPar:

secuencial	paralelo	aceleración	tamaño
22.4775	65.036	0.3456162741	2
1.3315	8.2377	0.161634922	4
7.9278	17.3337	0.457363402	8
23.3255	60.4964	0.385568397	16
23.0865	39.1417	0.589818531	32
78.0063	95.7016	0.815099225	64
418.8877	452.9224	0.924855339	128
763.8949	1643.1642	0.464892614	256
5255.3915	7082.2567	0.742050412	512
38241.9498	46519.9213	0.822055342	1024

mulMatrizRec y multMatrizRecPar:

secuencial	paralelo	aceleración	tamaño
16.421	160.4823	0.102322811	2
0.6665	3.6586	0.182173509	4
2.431	7.5121	0.323611241	8
5.5331	4.8079	1.150835084	16
9.9844	9.4263	1.059206688	32
114.009	76.1921	1.496336234	64
171.0772	110.2761	1.551353376	128
975.0616	349.0411	2.793543798	256
6816.132	2722.3861	2.5037345	512

Strassen secuencial y Strassen paralelo:

secuencial	paralelo	aceleración	tamaño
42.2317	46.7452	0.903444632	2
5.3831	7.5917	0.717584014	4
21.1837	46.2346	0.458178507	8
21.8697	54.7775	0.399246041	16
70.961	117.3424	0.604734521	32
414.6203	219.8303	1.886092591	64
1785.4567	1465.9401	1.217960202	128
9691.7861	4559.9461	2.125416811	256
69967.0796	32028.6872	2.184512876	512
455716.5819	222010.431	2.052680948	1024

Análisis comparativo de las funciones

- **Comparación entre las funciones secuenciales:**

  Como se puede ver en las tablas de resultados de todas las funciones secuenciales, la que obtuvo un mejor rendimiento fue la de multMatrizRec debido a que divide la matriz en submatrices.

- **Comparación entre las funciones paralelas:**

  Al igual que se puede ver en su versión secuencial, la versión paralela de esta (multMatrizRecPar) es la que obtiene un mejor rendimiento en comparación las demás funciones paralelas, ya que a parte de dividir en submatrices divide esas submatrices en subprocesos.

- **¿Las paralelizaciones sirvieron?**

  En general, la paralelización si sirvió, sin embargo, se presentan ciertas situaciones que de las cuales suele ser mejor el método secuencial dado que hay ciertos tamaños de matrices que trabajan de una manera más eficiente de este modo, como lo son las de un tamaño pequeño.

- **¿Es realmente más eficiente el algoritmo de Strassen?**

  Como se puede apreciar en los resultados, el algoritmo de Strassen presenta unos tiempos bastante elevados en comparación con las demás funciones, por lo que se puede concluir que no es muy eficiente.

- **¿No se puede concluir nada al respecto?**

  Se puede concluir que dependiendo de que es lo que queramos hacer y que tan grande sea, la paralelización puede llegar a ser un método de optimización.

Preguntas

- **¿Cuál de las implementaciones es más rápida?**

  La implementación más rápida a comparación de las demás funciones es multMatrizRecPar.

- **¿De qué depende que la aceleración sea mejor?**

  La aceleración depende de los tiempos que se obtuvieron en cada uno de los tamaños correspondientes de las funciones secuenciales y paralelas, entre más alto sea el tiempo de la versión secuencial y entre menor sea el tiempo de la versión paralela, el cociente (aceleración) será mejor.

- **¿Puede caracterizar los casos en que es mejor usar la versión secuencial/paralela de cada algoritmo de multiplicación de matrices?**

  Se puede apreciar que entre mayor sea el tamaño de la matriz es mejor usar la versión paralela, y entre más pequeña sea es mejor la versión secuencial.