Universidad del Valle

Facultad de ingeniería

Ingeniería en sistemas

Cristian David Pacheco Torres
2227437
Juan Sebastian Molina Cuellar
2224491

November 9, 2023

Taller 4: Colecciones y Expresiones For:
El problema de la subsecuencia incremental de longitud máxima

Contents

1 Introducción

1.1 Preliminares

Con el proposito de implementar diferentes algoritmos de multiplicación de matrices, tanto secuenciales, recursivos y paralelos, se nos otorgó a través del *Taller 5: Multiplicación de matrices en paralelo*, la definicion matemática de las siguientes operaciones:

- Transpuesta de una matriz.

$$T : \mathbb{R}^{m\times n} \to \mathbb{R}^{n\times m}$$
$$T(A) = [a_{ji}]_{n\ \times\ m} \tag{1}$$

  Se denota como : $\mathbf{A}^T$

- Producto punto de vectores.

$$\cdot : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$$
$$\mathbf{u} \cdot \mathbf{v} = \sum_{i=1}^{n} u_i\ v_i \tag{2}$$

  Donde "$\cdot$" denota la operacion binaria entre dos vectores $\in \mathbb{R}^n$.

- Multiplicación de matrices.

$$C_{ij} = \sum_{k=1}^{n} A_{ik} \cdot\ B_{kj} \tag{3}$$

- Suma de matrices.

$$+ : \mathbb{R}^{m\times n} \times \mathbb{R}^{m\times n} \to \mathbb{R}^{m\times\ n}$$
$$A\ +\ B\ =\ C$$
$$C_{ij} = A_{ij} + B_{ij} \tag{4}$$

  Donde "+" representa la operación de adición entre dos matrices $\in \mathbb{R}^{m\times n}$. A y B son las matrices a sumar. C es la matriz resultante de la suma.

- Resta de matrices.

$$- : \mathbb{R}^{m\times n} \times \mathbb{R}^{m\times n} \to \mathbb{R}^{m\times\ n}$$
$$C_{ij} = A_{ij} - B_{ij} \tag{5}$$
$$A\ -\ B\ =\ C$$

Definiciones de utilidad:

Sea una tarea computacional $T = (t, r)$, donde t es el tiempo de ejecución y r el resultado.
Además definase dos funciones sobre T como:

$$\rho(T) = \rho(t, r) = r \tag{6}$$

$$\phi(T) = \phi(t, r) = t \tag{7}$$

Definase una computación secuencial S:

$$S =< T_1, T_2, ...T_i, ..., T_{n-1}, T_n > \tag{8}$$

Donde i representa el orden de ejecución de la tarea $T_i \mid 0 \leq i \leq n$.

Definase una computación paralela P:

$$P = \{T_1, T_2, ...T_i, ..., T_{n-1}, T_n\} \tag{9}$$

Donde i identifica cada tarea que a posteriori se le extraerá el tiempo de ejecución y el resultado.

Sea $\phi(S)$ el tiempo de ejecución de una secuencia S:

$$\phi(S) = \sum_{1}^{n} \phi(T_i) \tag{10}$$

Sea $\phi(T)$ el tiempo de ejecución del conjunto de tareas P:

$$\phi(P) = \max(\phi_i(T_i)) \mid 1 \leq i \leq n \wedge \phi_i(T_i) \in P \tag{11}$$

1.2 Algoritmos de proporcionados de utilidad

En el listing 1, se definen dos tipos de datos esenciales: `Matriz` y `MatrizD`. Estos tipos representan matrices de enteros, donde `MatrizD` está diseñada para un procesamiento paralelo en base al tipo $ParVector$ de $Scala$.

```scala
type Matriz = Vector[Vector[Int]]
type MatrizD = ParVector[ParVector[Int]]
```

Listing 1: Definiciones tipos de datos

```scala
def matrizAlAzar(long:Int, vals:Int) = {
    //Crea una matriz de enteros cuadrada de long x long,
    //con valores entre 0 y vals
    val v = Vector.fill(long, long){random.nextInt(vals)}
    v
}
```

Listing 2: matriz al azar

```scala
def vectorAlAzar(long:Int, vals:Int): Vector[Int] = {
    //Crea un vector de enteros de longitud long,
    //con valores aleatorios entre 0 y vals
    val v = Vector.fill(long){random.nextInt(vals)}
    v
}
```

Listing 3: vector al azar

```scala
def prodPunto(v1: Vector[Int], v2: Vector[Int]): Int = {
    //Calcula el producto punto entre dos vectores
    (v1 zip v2).map({case (i,j) => i*j}).sum
}
```

Listing 4: producto punto

```scala
def transpuesta(m: Matriz): Matriz = {
    //Calcula la transpuesta de una matriz
    val l =m.length
    Vector.tabulate(l,l)((i,j)=>m(j)(i))
}
```

Listing 5: transpuesta de una matriz

Los algoritmos anteriores fueron sugeridos en el $Taller5$ por parte del profesor, para la implementación de las funciones a desarrollar en este informe. Estos algoritmos fueron de utilidad para generar matrices aletorias y operaciones fundamentales entre vectores.

2 Informe del taller - secciones

2.1 Informe de corrección

multMatriz

```scala
def multMatriz(m1: Matriz, m2: Matriz): Matriz = {
    val l = m1.length
    val m = m2.length
    val v =Vector.tabulate(l,m)((i,j)=>prodPunto(m1(i),
       transpuesta(m2)(j)))
    v
}
```

Listing 6: mult matriz

multMatrizPar

```scala
def multMatrizPar(m1: Matriz, m2: Matriz): Matriz = {
    val l = m1.length
    val m = m2.length
    val parRows = for (k <- 0 until l )
        yield {
            (k,task(Vector.tabulate(1,m)((i,j)=>
                prodPunto(m1(k),transpuesta(m2)(j)))))
        }
    val v = parRows.map({case (i,j) =>
        (i,j.join())}).sortBy(_._1).map(_._2)
    v.reduce(_++_)
```

```
}
```

Listing 7: mult matriz paralela

multMatrizRec

```
def multMatrizRec(m1:Matriz, m2:Matriz): Matriz ={
    //recibe m1 y m2 matrices cuadradas de la misma dimension,
       potencia de 2
    //y devuelve la multiplicacion de las 2 matrices
    val n = m1.length
    if(n == 1) {
        Vector(Vector(m1(0)(0)*m2(0)(0)))
    }
    else {
        val l = n/2
        val (m1_11, m1_12, m1_21, m1_22) =
            (subMatriz(m1,0,0,l),subMatriz(m1,0,l,l),
             subMatriz(m1,l,0,l),subMatriz(m1,l,l,l))
        val (m2_11, m2_12, m2_21, m2_22) =
            (subMatriz(m2,0,0,l),subMatriz(m2,0,l,l),
             subMatriz(m2,l,0,l),subMatriz(m2,l,l,l))

        val c_11 = sumMatriz(multMatrizRec(m1_11,m2_11),
           multMatrizRec(m1_12,m2_21))
        val c_12 = sumMatriz(multMatrizRec(m1_11,m2_12),
           multMatrizRec(m1_12,m2_22))
        val c_21 = sumMatriz(multMatrizRec(m1_21,m2_11),
           multMatrizRec(m1_22,m2_21))
        val c_22 = sumMatriz(multMatrizRec(m1_21,m2_12),
           multMatrizRec(m1_22,m2_22))

        Vector.tabulate(n,n)((i,j)=>
            if(i<l && j<l) c_11(i)(j)
            else if(i<l && j>=l) c_12(i)(j-l)
            else if(i>=l && j<l) c_21(i-l)(j)
            else c_22(i-l)(j-l))
    }
}
```

Listing 8: mult matriz recursiva

multMatrizRecPar

```
def multMatrizRecPar(m1:Matriz, m2:Matriz): Matriz ={
    //recibe m1 y m2 matrices cuadradas de la misma dimension,
       potencia de 2
    //y devuelve la multiplicacion de las 2 matrices,
       paralelizando tareas
    val n = m1.length
```

```
    if(n == 1) {
        Vector(Vector(m1(0)(0)*m2(0)(0)))
    }
    else {
        val l = n/2
        val (m1_11, m1_12, m1_21, m1_22) =
            (subMatriz(m1,0,0,l),subMatriz(m1,0,l,l),
             subMatriz(m1,l,0,l),subMatriz(m1,l,l,l))
        val (m2_11, m2_12, m2_21, m2_22) =
            (subMatriz(m2,0,0,l),subMatriz(m2,0,l,l),
             subMatriz(m2,l,0,l),subMatriz(m2,l,l,l))
        val (c_11, c_12, c_21, c_22) = parallel(
            sumMatriz(multMatrizRec(m1_11,m2_11),
                      multMatrizRec(m1_12,m2_21)),
            sumMatriz(multMatrizRec(m1_11,m2_12),
                      multMatrizRec(m1_12,m2_22)),
            sumMatriz(multMatrizRec(m1_21,m2_11),
                      multMatrizRec(m1_22,m2_21)),
            sumMatriz(multMatrizRec(m1_21,m2_12),
                      multMatrizRec(m1_22,m2_22)))
        Vector.tabulate(n,n)((i,j)=>
            if(i<l && j<l) c_11(i)(j)
            else if(i<l && j>=l) c_12(i)(j-l)
            else if(i>=l && j<l) c_21(i-l)(j)
            else c_22(i-l)(j-l))
    }
}
```

Listing 9: mult matriz recursiva paralela

multStrassen

```
def multStrassen(m1:Matriz, m2:Matriz): Matriz ={
    //recibe m1 y m2 matrices cuadradas de la misma dimension, potencia de 2
    //y devuelve la multiplicacion de las 2 matrices
    val n = m1.length
    if(n == 1) {
        Vector(Vector(m1(0)(0)*m2(0)(0)))
    }
    else {
        val l = n/2
        val (s1, s2, s3, s4, s5, s6, s7, s8, s9, s10) = (
            restaMatriz(subMatriz(m2,0,l,l),subMatriz(m1,l,l,l)),
            sumMatriz(subMatriz(m1,0,0,l), subMatriz(m1,0,l,l)),
            sumMatriz(subMatriz(m1,l,0,l), subMatriz(m1,l,l,l)),
            restaMatriz(subMatriz(m2,l,0,l), subMatriz(m2,0,0,l)),
            sumMatriz(subMatriz(m1,0,0,l), subMatriz(m1,l,l,l)),
            sumMatriz(subMatriz(m2,0,0,l), subMatriz(m2,l,l,l)),
```

```
            restaMatriz(subMatriz(m1,0,l,l), subMatriz(m1,l,l,l))
                ,
            sumMatriz(subMatriz(m2,l,0,l), subMatriz(m2,l,l,l)),
            restaMatriz(subMatriz(m1,0,0,l), subMatriz(m1,l,0,l))
                ,
            sumMatriz(subMatriz(m2,0,0,l), subMatriz(m2,0,l,l))
        )
        val (p1, p2, p3, p4, p5, p6, p7) = (
            multStrassen(subMatriz(m1,0,0,l), s1),
            multStrassen(s2, subMatriz(m2,l,l,l)),
            multStrassen(s3, subMatriz(m2,0,0,l)),
            multStrassen(subMatriz(m1,l,l,l), s4),
            multStrassen(s5, s6),
            multStrassen(s7, s8),
            multStrassen(s9, s10)
        )
        val (c_11, c_12, c_21, c_22) = (
            restaMatriz(sumMatriz(p5, p4), sumMatriz(p6, p2)),
            sumMatriz(p1, p2),
            sumMatriz(p3, p4),
            restaMatriz(sumMatriz(p1, p5), restaMatriz(p3, p7))
        )
        Vector.tabulate(n,n)((i,j)=>
            if(i<l && j<l) c_11(i)(j)
            else if(i<l && j>=l) c_12(i)(j-l)
            else if(i>=l && j<l) c_21(i-l)(j)
            else c_22(i-l)(j-l))
    }
}
```

Listing 10: mult Strassen

multStrassenPar

```
    def multStrassenPar(m1:Matriz, m2:Matriz): Matriz ={
        //recibe m1 y m2 matrices cuadradas de la misma dimension
            , potencia de 2
        //y devuelve la multiplicacion de las 2 matrices
        val n = m1.length

        /*if(umbral <= n){
            multMatrizRec(m1,m2)
        }*/
        if(n == 1) {
            Vector(Vector(m1(0)(0)*m2(0)(0)))
        }
        else {
            val l = n/2
            val (s1, s2, s3, s4, s5, s6, s7, s8, s9, s10) = (
                restaMatriz(subMatriz(m2,0,l,l),
                            subMatriz(m1,l,l,l)),
```

```scala
          sumMatriz(subMatriz(m1,0,0,l),
               subMatriz(m1,0,l,l)),
          sumMatriz(subMatriz(m1,l,0,l),
               subMatriz(m1,l,l,l)),
          restaMatriz(subMatriz(m2,l,0,l),
                subMatriz(m2,0,0,l)),
          sumMatriz(subMatriz(m1,0,0,l),
               subMatriz(m1,l,l,l)),
          sumMatriz(subMatriz(m2,0,0,l),
               subMatriz(m2,l,l,l)),
          restaMatriz(subMatriz(m1,0,l,l),
                subMatriz(m1,l,l,l)),
          sumMatriz(subMatriz(m2,l,0,l),
               subMatriz(m2,l,l,l)),
          restaMatriz(subMatriz(m1,0,0,l),
                subMatriz(m1,l,0,l)),
          sumMatriz(subMatriz(m2,0,0,l),
               subMatriz(m2,0,l,l))
        )
        val (p1, p2, p3, p4, p5, p6, p7) = (
          task(multStrassenPar(subMatriz(m1,0,0,l), s1)),
          task(multStrassenPar(s2, subMatriz(m2,l,l,l))),
          task(multStrassenPar(s3, subMatriz(m2,0,0,l))),
          task(multStrassenPar(subMatriz(m1,l,l,l), s4)),
          task(multStrassenPar(s5, s6)),
          task(multStrassenPar(s7, s8)),
          task(multStrassenPar(s9, s10))
        )
        val (c_11, c_12, c_21, c_22) = (
          restaMatriz(sumMatriz(p5.join(), p4.join()),
            sumMatriz(p6.join(), p2.join())),
          sumMatriz(p1.join(),p2.join()),
          sumMatriz(p3.join(),p4.join()),
          restaMatriz(sumMatriz(p1.join(), p5.join()),
            restaMatriz(p3.join(), p7.join()))
        )
        Vector.tabulate(n,n)((i,j)=>
          if(i<l && j<l) c_11(i)(j)
          else if(i<l && j>=l) c_12(i)(j-l)
          else if(i>=l && j<l) c_21(i-l)(j)
          else c_22(i-l)(j-l))
      }
    }
```

Listing 11: mult Strassen paralela

2.1.1 Argumentación de Corrección

Argumentación detallada de la corrección para cada uno de los algoritmos de multiplicación de matrices y productos punto.

2.1.2 Explicación Teórica y Método de Inducción

Explicación teórica de la corrección y método de inducción o inducción estructural utilizado.

2.1.3 Conclusión de Corrección

Conclusión sobre la corrección de cada función.

2.2 Informe de desempeño de las funciones secuenciales y de las funciones paralelas

2.2.1 Resultados de Multiplicación de Matrices

Tabla comparativa y análisis de desempeño.

2.2.2 Metodología de Generación de Matrices de Prueba

Descripción de cómo se generaron las matrices de prueba.

2.2.3 Análisis de Resultados

Análisis profundo de los resultados obtenidos.

2.2.4 Resultados de Producto Punto de Vectores

Tabla comparativa y análisis de desempeño para las implementaciones de producto punto.

2.2.5 Impacto de las Dimensiones de los Vectores

Discusión sobre cómo las dimensiones de los vectores afectan al desempeño.

2.2.6 Análisis de Resultados del Producto Punto

Análisis detallado de los resultados del producto punto.

2.3 Análisis comparativo de las diferentes soluciones

2.3.1 Análisis Basado en Resultados

Comparación entre las versiones secuenciales y paralelas de los algoritmos y su desempeño.

2.3.2 Eficiencia del Algoritmo de Strassen

Discusión específica sobre la eficiencia del algoritmo de Strassen en comparación con otros.

2.3.3 Reflexiones sobre el Paralelismo

Evaluación crítica del paralelismo de tareas y de datos y su efecto en la eficiencia general.

3 Conclusiones

3.1 Síntesis de Hallazgos

Resumen de los hallazgos más importantes del informe.

3.2 Implicaciones de los Resultados

Discusión sobre las implicaciones de los resultados para futuras investigaciones y aplicaciones prácticas.

3.3 Recomendaciones

Recomendaciones basadas en el análisis y desempeño de las funciones estudiadas.