1. **Analiza Cerințelor**

**Tema:** Implementarea unui algoritm de convoluție folosind programarea paralelă în Java.

**Cerințe:** Utilizarea diferitelor strategii de partiționare a datelor (pe verticala, pe orizontala, pe blocuri).

1. **Proiectare**

**Structuri de Date:**

int[][] F: Matricea initiala

int[][] F\_padded: Matricea de intrare extinsă cu padding.

int[][] C: Matricea de convoluție.

int[][] V: Matricea rezultată după aplicarea convoluției.

int[][] matrixThread: Matrice pentru a stoca numărul thread-ului care a procesat fiecare element.

**Partiționare pe Threaduri:**

Orizontala: Fiecare thread procesează un set de coloane.

Verticală: Fiecare thread procesează un set de randuri.

Pe Blocuri: Fiecare thread procesează un bloc de elemente.

**Clase:**

*Secvential:* Clasa pentru calcularea convolutiei intr un mod secvential

*Convolutie*: Clasa principală care conține metodele de convoluție.

*ConvolutieUtils:* Clasa care stocheaza metoda de bordare

*ThreadVerticala*: Clasa pentru thread-urile care procesează randuri.

*ThreadOrizontala*: Clasa pentru thread-urile care procesează coloane.

*ThreadBloc*: Clasa pentru thread-urile care procesează blocuri.

**Funcții:**

*convolutieSecventiala:* Medota pentru convolutia secventiala.

convolutieVerticala: Metodă pentru convoluția verticală.

convolutieOrizontala: Metodă pentru convoluția orizontala.

convolutieBloc: Metodă pentru convoluția pe blocuri.

1. **Detalii de Implementare**

**Convolutie Secventiala:**

public static int[][] convolutieSecventiala(int[][] F\_padded, int[][] C, int n, int m, int k) {}

**Convoluția Verticală:**

public static int[][] convolutieVerticala(int[][] F\_padded, int[][] C, int n, int m, int numThreads) {}

**Convoluția Orizontala:**

public static int[][] convolutieOrizontala(int[][] F\_padded, int[][] C, int n, int m, int numThreads){}

**Convoluția pe Blocuri:**

public static int[][] convolutieBloc(int[][] F\_padded, int[][] C, int n, int m, int numThreads) {}

1. **Cazuri de Testare**

Toate tipurile de distributie (pe verticala, pe orizontala, pe blocuri):

- 4 thread-uri pe o matrice de 10x10 si filtru de 3X3.

- 2/4/8/16 thread-uri pe o matrice 1000X1000 si filtru de 5X5

- 2/4/8/16 thread-uri pe o matrice 10X1000 si filtru de 5X5

- 2/4/8/16 thread-uri pe o matrice 1000X10 si filtru de 5X5

- 2/4/8/16 thread-uri pe o matrice 10000X10000 si filtru de 5X5

**Concluzii:**

* C++ static are cei mai buni timpi comparative cu C++ dynamic si Java. Totusi intre C++ si Java, C++ are timpul cel mai bun.