Tema 1

Enunt:

Se considera o imagine reprezentata printr-o matrice de pixeli, F , de dimensiune (MxN).

Se cere transformarea ei aplicand o filtrare cu o fereastra definita de multimea de indici

Ind[n,m] = {(k,l) | -n/2<=k<=n/2, -m/2<=l<=m/2}

si de coeficientii wkl

n,m impare si n<N, m<M.

Transformarea unui pixel:

V[i,j] = {+ (k,l): -n/2<=k<=n/2, -m/2<=l<=m/2: w[k,l] \* F[i-k ,j-l] }

unde

De exemplu:

multimea de indici este

Ind [3,3] ={ (-1,-1), (-1,0), (-1,1), (0,-1), (0,0), (-0,1), (1,-1), (1,0), (1,1)}

Si ponderile asociate

Actualizarea unui pixel de pe pozitia (i,j)

v[i,j] =f[i,j] \*1/9+f[i-1,j]\* 1/9+f[i,j-1]\* 1/9+f[i-1,j-1]\* 1/9+f[i+1,j]\* 1/9+f[i,j+1]\* 1/9+f[i+1,j+1]\* 1/9

Pentru frontiere se considera ca un element este egal cu elemental din celula vecina din matrice f[-1,-1]= f[0,0]; f[-1,j]= f[0,j]; f[i,-1]=f[i,0]; f[M,j]= f[M-1,j]; f[i,N]=f[i,N-1];

Exemplificare -> https://de.wikipedia.org/wiki/Datei:2D\_Convolution\_Animation.gif

Se cere asigurarea urmatoarei postconditii:

Postconditie: Matricea rezultat V contine imaginea filtrata a imaginii initiale F (unde V != F)

A) Program secvential

B) Program paralel: folositi p threaduri pentru calcul.

Obiectiv: Impartire cat mai echilibrata si eficienta a calculul pe threaduri!

Pentru impartirea sarcinilor de calcul (taskuri) se foloseste descompunere geometrica care poate fi (puteti alege o varianta sau sa incercati mai multe si sa o identificati pe cea mai buna):

- Pe orizontala (mai multe linii alocate unui thread)

- Pe verticala (mai multe coloane alocate unui thread)

- Bloc – submatrici alocate unui thread

- bazat pe o functie de distributie prin care unui index al unui thread i se distribuie o submultime de indecsi din matrice;

distributia se poate face prin:

- distributie liniara (indici alaturati la acelasi thread) sau

- distributie ciclica( cu pas egal cu p).

Datele de intrare se citesc dintr-un fisier de intrare “date.txt”.

(Fisierul trebuie creat anterior prin adaugare de numere generate aleator. Toate rularile trebuie executate cu acelasi fisier.)

Implementare:

a) Java

b) C++ ( cel putin C++11 )

i. matricile sunt alocate static (int f[MAX][MAX] )

ii. matricile sunt alocate dynamic (new…)

Folosire directa a threadurilor (creare explicita) => nu se permite folosirea executorilor.

Testare: masurati timpul de executie pentru

1) N=M=10 si n=m=3; p=4;

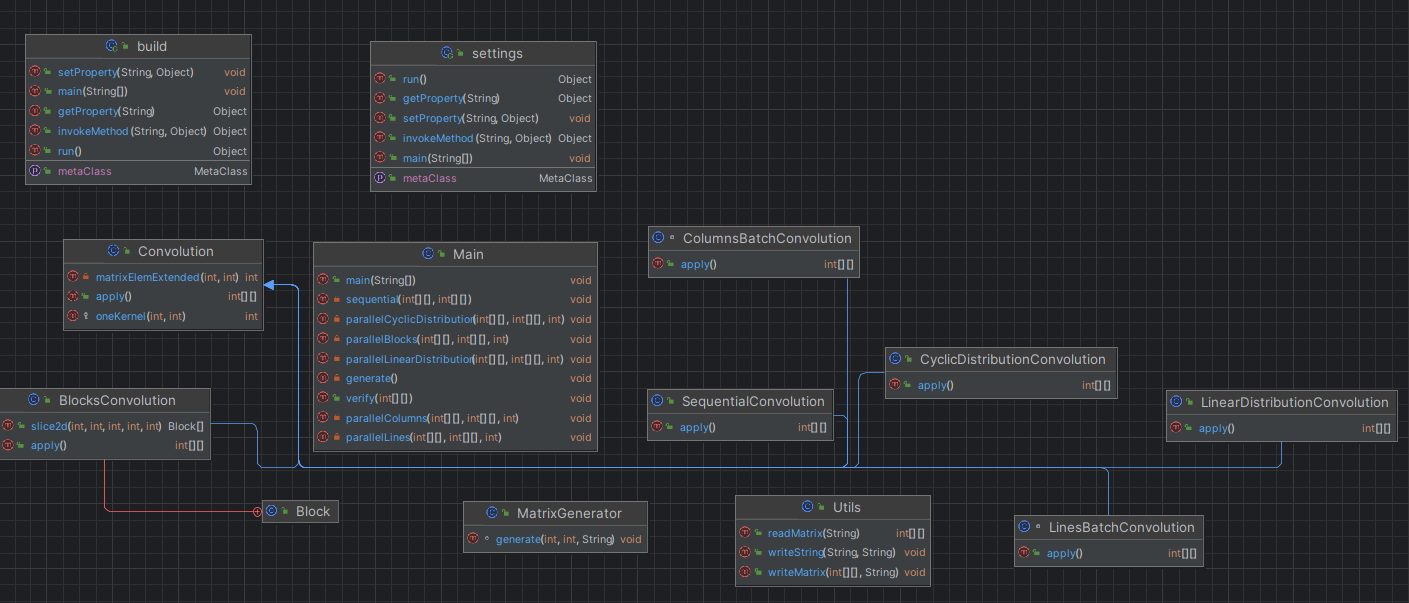
2) N=M=1000 si n=m=5; p=2,4,8,16

3) N=10 M=10000 si n=m=5; p=2,4,8,16

4) N=10000 M=10 si n=m=5; p=2,4,8,16

5) N=10000 M=10 si n=m=5; p=2,4,8,16

Java:



C++:

|  |
| --- |
| Program - Static |
| N: Int  M: Int  n: Int  p: Int  mat: Int[][]  filter: Int[][]  result: Int[][] |
| readImage(): void  readKernel(): void  verify(): bool  matrixElemExtended(int i, int j): int  oneKernel(int i, int j): int  runSequentiallyStatic(): void  workerLines(int start, int end): void  runParallelStaticLines(): void  workerColumns(int start, int end): void  runParallelStaticColumns(): void  struct Block { int firstLine, firstColumn, lastLine, lastColumn}  slice2d(int i, int j, int n, int m, int p, Block blocks[], int \*blockIndex): Block\*  workerBlock(Block block): void  runParralelStaticBlocks(): void  workerLinear(): void  runParallelStaticLinear(): void  workerCyclic(): void  runParallelStaticCyclic(): void  main(int argc, char\*\* argv): int |

|  |
| --- |
| Program - Dinamic |
| N: Int  M: Int  n: Int  p: Int  mat2: Int\*\*  filter2: Int\*\*  result2: Int\*\* |
| readImage(): void  readKernel(): void  verify(): bool  matrixElemExtended(int i, int j): int  oneKernel(int i, int j): int  runSequentiallyDynamic(): void  workerLines(int start, int end): void  runParallelDynamicLines(): void  workerColumns(int start, int end): void  runParallelDynamicColumns(): void  struct Block { int firstLine, firstColumn, lastLine, lastColumn}  slice2d(int i, int j, int n, int m, int p, Block blocks[], int \*blockIndex): Block\*  workerBlock(Block block): void  runParralelDynamicBlocks(): void  workerLinear(): void  runParallelDynamicLinear(): void  workerCyclic(): void  runParallelDynamicCyclic(): void  main(int argc, char\*\* argv): int |

Partiționarea datelor

* Matricea citita din fisier ramane nemodificata
* Thread primește un start, end
* Thread-ul calculează valoarea elementelor de pe liniile sau coloanele de la start pana la end
* Rezultatele se stocheaza intr-o matrice noua
* Ne asiguram ca diferenta dintre numarul elementelor dintre doua treduri nu este mai mare decat 1, distribuind elementele ap. egal intre threaduri:

Ex:

int columnsPerThread = M / P;  
int columnsLeft = M % P;

for (int k = 0; k < M; k += columnsPerThread) {

if (columnsLeft > 0) {  
 columnsLeft--;  
 k++;  
}

…

}

|  |  |  |  |
| --- | --- | --- | --- |
| Matrice | Nr. Threaduri | Algoritm | Timp |
| N=M=10 n=m=3 | 1 | Secvențial | 16.28436 |
| 4 | Linii | 13.1899 |
| Coloane | 14.56824 |
| Blocuri | 20.54228 |
| Distibuție Liniară | 21.273 |
| Distribuție Ciclică | 13.96329 |
| N=M=1000 si n=m=5 | 1 | Secvențial | 961.14095 |
| 2 | Linii | 1018.68826 |
| Coloane | 1076.95326 |
| Blocuri | 1171.48189 |
| Distibuție Liniară | 1045.82773 |
| Distribuție Ciclică | 988.47242 |
| 4 | Linii | 1075.96248 |
| Coloane | 985.36237 |
| Blocuri | 1015.22547 |
| Distibuție Liniară | 1032.72653 |
| Distribuție Ciclică | 1011.60387 |
| 8 | Linii | 1066.15517 |
| Coloane | 1133.1928 |
| Blocuri | 1053.29896 |
| Distibuție Liniară | 1006.12716 |
| Distribuție Ciclică | 1028.26752 |
| 16 | Linii | 1083.21578 |
| Coloane | 1121.91383 |
| Blocuri | 1063.854 |
| Distibuție Liniară | 1200.03113 |
| Distribuție Ciclică | 1207.46335 |
| N=10 M=10000 si n=m=5 | 1 | Secvențial | 128.96499 |
| 2 | Linii | 124.47681 |
| Coloane | 125.55024 |
| Blocuri | 126.7992 |
| Distibuție Liniară | 128.96732 |
| Distribuție Ciclică | 130.56032 |
| 4 | Linii | 133.14402 |
| Coloane | 132.41665 |
| Blocuri | 128.35066 |
| Distibuție Liniară | 129.35342 |
| Distribuție Ciclică | 123.92209 |
| 8 | Linii | 130.55031 |
| Coloane | 134.18304 |
| Blocuri | 126.02883 |
| Distibuție Liniară | 135.73681 |
| Distribuție Ciclică | 132.03652 |
| 16 | Linii | 126.09129 |
| Coloane | 128.05487 |
| Blocuri | 127.28138 |
| Distibuție Liniară | 127.63595 |
| Distribuție Ciclică | 127.3638 |
| N=10000 M=10 si n=m=5 | 1 | Secvențial | 132.28599 |
| 2 | Linii | 128.78543 |
| Coloane | 130.7789 |
| Blocuri | 126.44326 |
| Distibuție Liniară | 128.07369 |
| Distribuție Ciclică | 129.87765 |
| 4 | Linii | 130.45987 |
| Coloane | 129.0023 |
| Blocuri | 130.74543 |
| Distibuție Liniară | 133.55011 |
| Distribuție Ciclică | 131.34446 |
| 8 | Linii | 129.0453 |
| Coloane | 128.56988 |
| Blocuri | 129.77989 |
| Distibuție Liniară | 132.4432 |
| Distribuție Ciclică | 133.59087 |
| 16 | Linii | 130.28083 |
| Coloane | 129.19058 |
| Blocuri | 132.20124 |
| Distibuție Liniară | 129.05123 |
| Distribuție Ciclică | 131.11036 |
| N=10000 M=10000 si n=m=5; | 1 | Secvențial | 129921.93644 |
| 2 | Linii | 99085.82318 |
| Coloane | 129655.6842 |
| Blocuri | 71182.6284666667 |
| Distibuție Liniară | / |
| Distribuție Ciclică | / |
| 4 | Linii | 106731.90988 |
| Coloane | 73402.158 |
| Blocuri | 98404.82045 |
| Distibuție Liniară | / |
| Distribuție Ciclică | / |
| 8 | Linii | 108561.801766667 |
| Coloane | 69843.4318 |
| Blocuri | / |
| Distibuție Liniară | / |
| Distribuție Ciclică | / |
| 16 | Linii | 106346.654766667 |
| Coloane | 73742.11605 |
| Blocuri | / |
| Distibuție Liniară | / |
| Distribuție Ciclică | / |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Matrice | Nr. Threaduri | Algoritm | Timp | | |
| Static | | Dinamic |
| N=M=10 n=m=3 | 1 | Secvențial | 0.19396 | | 0.24264 |
| 4 | Linii | 1.35755 | | 0.91318 |
| Coloane | 1.88495 | | 1.79345 |
| Blocuri | 1.94021 | | 1.64788 |
| Distibuție Liniară | 1.38901 | | 1.12945 |
| Distribuție Ciclică | 1.06738 | | 1.14078 |
| N=M=1000 si n=m=5 | 1 | Secvențial | 350.30863 | | 356.6936 |
| 2 | Linii | 322.8975 | | 275.6194 |
| Coloane | 297.6332 | | 333.1747 |
| Blocuri | 335.4623 | | 370.3556 |
| Distibuție Liniară | 338.9862 | | 460.5711 |
| Distribuție Ciclică | 278.9246 | | 295.2989 |
| 4 | Linii | 285.1357 | | 297.7117 |
| Coloane | 379.3089 | | 294.801 |
| Blocuri | 315.1412 | | 260.9067 |
| Distibuție Liniară | 275.5514 | | 284.2149 |
| Distribuție Ciclică | 272.2088 | | 279.2141 |
| 8 | Linii | 282.0556 | | 292.1262 |
| Coloane | 269.8484 | | 284.7412 |
| Blocuri | 290.782 | | 311.1939 |
| Distibuție Liniară | 283.0813 | | 292.349 |
| Distribuție Ciclică | 289.02 | | 326.9136 |
| 16 | Linii | 334.0161 | | 303.0514 |
| Coloane | 278.7822 | | 281.2576 |
| Blocuri | 280.5301 | | 296.1192 |
| Distibuție Liniară | 310.2423 | | 282.6328 |
| Distribuție Ciclică | 281.0085 | | 279.4984 |
| N=10 M=10000 si n=m=5 | 1 | Secvențial | 15.00829 | | 17.83561 |
| 2 | Linii | 14.18004 | | 15.63912 |
| Coloane | 15.00291 | | 15.64876 |
| Blocuri | 14.71861 | | 16.84646 |
| Distibuție Liniară | 15.46085 | | 16.01032 |
| Distribuție Ciclică | 14.64486 | | 14.84747 |
| 4 | Linii | 14.20577 | | 15.68762 |
| Coloane | 15.12949 | | 15.57228 |
| Blocuri | 13.78503 | | 15.75528 |
| Distibuție Liniară | 14.92315 | | 14.40987 |
| Distribuție Ciclică | 14.31726 | | 14.76607 |
| 8 | Linii | 14.2113 | | 15.49279 |
| Coloane | 14.13514 | | 15.82701 |
| Blocuri | 13.83599 | | 16.02653 |
| Distibuție Liniară | 14.95155 | | 14.13077 |
| Distribuție Ciclică | 15.17333 | | 15.52454 |
| 16 | Linii | 13.73998 | | 15.51637 |
| Coloane | 15.835 | | 15.82701 |
| Blocuri | 14.73768 | 16.65664 | |
| Distibuție Liniară | 14.61098 | 14.24016 | |
| Distribuție Ciclică | 15.59292 | 14.37702 | |
| N=10000 M=10 si n=m=5 | 1 | Secvențial | 14.876 | 17.76 | |
| 2 | Linii | 13.873 | 15.6777 | |
| Coloane | 14.9788 | 15.5443 | |
| Blocuri | 14.65 | 16.7879 | |
| Distibuție Liniară | 13.2389 | 14.0988 | |
| Distribuție Ciclică | 15.7776 | 15.9483 | |
| 4 | Linii | 14.7237 | 14.8873 | |
| Coloane | 15.4221 | 15.835 | |
| Blocuri | 15.8098 | 16.6558 | |
| Distibuție Liniară | 17.223 | 16.853 | |
| Distribuție Ciclică | 15.7533 | 15.60 | |
| 8 | Linii | 19.7662 | 15.9899 | |
| Coloane | 15.6454 | 15.8967 | |
| Blocuri | 14.224 | 15.9437 | |
| Distibuție Liniară | 16.8455 | 16.7732 | |
| Distribuție Ciclică | 15.9998 | 15.8791 | |
| 16 | Linii | 14.1123 | 15.023 | |
| Coloane | 17.883 | 15.6349 | |
| Blocuri | 15.2753 | 15.1121 | |
| Distibuție Liniară | 14.4497 | 15.0034 | |
| Distribuție Ciclică | 16.0122 | 15.9967 | |
| N=10000 M=10000 si n=m=5; | 1 | Secvențial | 39518.04 | 15444.59 | |
| 2 | Linii | 32047.175 | 14996.75 | |
| Coloane | 29549.3 | 13886 | |
| Blocuri | 29815.7 | 13934.7 | |
| Distibuție Liniară | 32335.6 | 13837.3 | |
| Distribuție Ciclică | 32335.6 | 13825.2 | |
| 4 | Linii | 33389 | 13269.4 | |
| Coloane | 35502.5 | 13283.65 | |
| Blocuri | 29282.9 | 13751.65 | |
| Distibuție Liniară |  | 14504.6 | |
| Distribuție Ciclică | 33278.4 | 13761.26 | |
| 8 | Linii | 26178.166 | 12953.6 | |
| Coloane | 26224.15 | 13403 | |
| Blocuri | 22675.9 | 12736.3 | |
| Distibuție Liniară | 24013.7 | 15394.7 | |
| Distribuție Ciclică | 30790.6 | 13379.11 | |
| 16 | Linii | 36464.4333333333 | 13140.1 | |
| Coloane | 28697 | 12888.4 | |
| Blocuri | 29870.4 | 14435 | |
| Distibuție Liniară | 30224.457 | 14297.2 | |
| Distribuție Ciclică | 31.2102 | 13772.68 | |

Concluzii

* implementarea în C++ s-a dovedit semnificativ mai rapidă.
* În Java, nu putem trage o concluzie clară: observăm că numărul optim de thread-uri variază în fiecare caz.
* In C++ alocare statică a dat cel mai bun timp pentru matrici mai mici