C++

|  |  |  |
| --- | --- | --- |
| TIP MATRICE | THREADS | TIMP EXECUTIE |
| 10x10 | secvential | 0.00913 |
|  | 2 | 0.89068 |
| 1000x1000 | secvential | 61.72708 |
|  | 2 | 40.98526 |
|  | 4 | 25.09992 |
|  | 8 | 22.34277 |
|  | 16 | 15.31637 |
| 10000x10000 | secvential | 6353.104 |
|  | 2 | 4214.562 |
|  | 4 | 2400.499 |
|  | 8 | 1569.965 |
|  | 16 | 1146.559 |

JAVA

|  |  |  |
| --- | --- | --- |
| TIP MATRICE | THREADS | TIMP EXECUTIE |
| 10x10 | secvential | 0.05824 |
|  | 2 | 1.32208 |
| 1000x1000 | secvential | 27.95752 |
|  | 2 | 20.9866 |
|  | 4 | 26.76316 |
|  | 8 | 43.86715 |
|  | 16 | 54.32794 |
| 10000x10000 | secvential | 1186.075 |
|  | 2 | 753.3657 |
|  | 4 | 420.9725 |
|  | 8 | 319.5678 |
|  | 16 | 269.6698 |

*Principiu Buffer*

* Cazul secvential se bazeaza pe principul de salvare in buffer a liniei precedente si a liniei curente, nefiind nevoie pentru urmatoarele linii sa se adauge, in cazul de thread-uri fiind necesar inca o linie in buffer, in care sa se mai adauge linia de dupa ultima linie calculata de thread-ul respectiv.

*C++ vs Java*

* In cazul matricei de 10x10, se poate observa ca C++ se descurca mult mai bine decat Java in ambele cazuri, cazul secvential avand un timp de 5 ori mai bun.
* In cazul matricei de 1000x1000, tot C++ se descurca mai bine decat java in cazul unui numar de thread-uri mai mare, insa executia secventiala din java este mult mai rapida decat cea din c++. Diferenta este ca cu cat crestem numarul de thread-uri, timpul de executie creste pentru Java, probabil din cauza faptului ca este mai costisitor creearea thread-urilor si pornirea lor, iar pentru C++ scade, el lucrand direct pe processor. Totusi interesant este in cazul a 2 thread-uri, java se descurca mai bine decat c++.
* In cazul matricei de 10000x10000, numarul de date alocate este mult mai mare. Aici se observa ca implementarea in java este cu mult mai avantajoasa decat cea in C++. Se poate observa ca performanta in java (secvential) este aproximativ la fel cu cea din c++ (16 threads), iar implementarea cu 16 thread-uri din java este de ~5 ori mai rapida decat cea din c++. De asemenea, este de mentionat faptul ca cu cat se creste numarul de thread-uri in Java, scade si timpul de executie, contrar matricei de 1000x1000.

*Complexitatea de spatiu*

* Vom stoca matricea mare de dimensiuni n x n => O(n^2) pentru matricea pe care se aplica convolutia.
* Pentru matricea de convolutie care este 3x3, dimensiunea ei este insignifianta pentru calculul nostru.
* Fiecare thread va aloca un buffer pentru el insusi astfel incat sa fie posibil calculul convolutiei direct in matricea existenta. Un buffer va avea 3 coloane mereu. => O(n) este complexitatea de spatiu a unui buffer.
* Fiecare thread va primi size=n/noOfThreads[+ 1] numar de linii pe care va aplica convolutia.
* Complexitatea de spatiu finala este: O(n^2+threads\*n)⬄O(n^2)