Documentatie lab2

Oniga Andreea-Simona – 235/1

Java:

|  |  |  |
| --- | --- | --- |
| **Tip matrice** | **Nr. threads** | **Timp executie** |
| N=M=10  n=m=3 | Secvential | 0.4 |
| 4 | 2.9 |
| N=M=1000  n=m=3 | Secvential | 28.875 |
| 2 | 17 |
| 4 | 19.22 |
| 8 | 24 |
| 16 | 21.33 |
| N=10000  M=10000  n=m=3 | Secvential | 2704 |
| 2 | 2290 |
| 4 | 1019 |
| 8 | 674 |
| 16 | 645.5 |

C++

|  |  |  |
| --- | --- | --- |
| **Tip matrice** | **Nr. threads** | **Timp executie** |
| N=M=10  n=m=3 | Secvential | 0.0456 |
| 4 | 7.1087 |
| N=M=1000  n=m=3 | Secvential | 318.662 |
| 2 | 171.46 |
| 4 | 99.483 |
| 8 | 84.661 |
| 16 | 80.677 |
| N=10000  M=10000  n=m=3 | Secvential |  |
| 2 |  |
| 4 |  |
| 8 |  |
| 16 |  |

* Primul lucru pe care l-am observat este ca Java obtine timpi mult mai buni, desi implementarea este aceeasi
* In Java am reusit sa testez si cazul cu foarte multe date, in C++ insa nu, nu reusea sa-mi citeasca datele pentru a le putea prelucra
* In C++ putem observa ca eficienta creste cu cresterea numarului de thread-uri, in Java acest lucru este mult mai subtil
* Se mai observa ca timpii sunt putin mai buni decat laboratorul anterior

Metoda pe care am abordat-o pentru varianta secventiala, pentru a nu utiliza o matrice rezultat, a fost sa folosesc 2 vectori auxiliari care sa retina rezultatele din urma aplicarii convolutiei pe linia curenta si linia anterioara acesteia. Acesti 2 vectori au rolul de a ajuta ca se mentina datele initiale(necorupte) necesare aplicarii convolutiei.

Se calculeaza prima linie si se retin rezultatele in primul vector, apoi pentru fiecare din liniile care urmeaza se aplica convolutia pentru linia respectiva, apoi in matricea initiala se actualizeaza linia anterioara(din moment ce nu mai avem nevoie de ea), valorile liniei curente se salveaza in primul vector si se trece la urmatoarea linie.

Ultima linie va avea rezultatul aplicarii convolutiei in cel de al doilea vector, deci este important sa nu uitam sa actualizam si aceasta linie in matrice.

Pentru varianta paralela am abordat o metoda safe, dar nu foarte eficienta, prin care pentru fiecare thread voi avea un buffer care contine atat linile de la start la end cat si bordarea acestor linii. In urma prezentarii temei am inteles de ce nu este foarte eficienta abordarea mea, deoarece retin mai multe linii, cand as putea sa ma limitez la doar 3.

Pentru fiecare thread creez un buffer dupa cum am mentionat anterior si folosesc o bariera care asteapta ca toate thread-urile sa-si faca buffer-ul lor si abia apoi sa inceapa sa faca modificarile, preluand date din buffer si facand modificarile direct in matrice.