**Documentatie – Laborator 3**

Run option:

0 – secvential:

* Se parcurge matricea secvential linie cu linie, coloana cu coloana.

1 – impartire pe linii

* Fiecare thread primeste initial un numar de linii consecutive pe care le va actualiza egal cu n / p. Daca aceasta impartire are un rest, atunci primele n % p threaduri vor primi o linie in plus.

**Testare:**

**Testare C++:**

|  |  |  |  |
| --- | --- | --- | --- |
| Tip Matrice | Nr procese | Timp | Timp executie |
| 1000x1000 | 5 | T1 | 1.777 |
| 9 | 1.467 |
| 21 | 0.925 |
| 5 | T2 | 0.169 |
| 9 | 0.136 |
| 21 | 0.124 |

**Analiza:**

*Testarile s-au realizat pe un CPU Intel I3 12100F 4 cores 8 threads.*

* **C++** – numar diferit de procese

Se poate observa ca odata cu cresterea numarului de procese timpul scade semnificativ. Acest lucra era de asteptat, deoarece se micsoreaza numarul de elemente pe care fiecare proces trebuie sa il gestioneze.

* **C++** – laboratorul anterior

Comparativ cu timpii obtinuti la laboratorul anterior, timpii obtinuti la executia cu procese sunt mult mai mici, fapt ce se poate datora si modului de initializare a cache-ul fiecare proces putand incepe executia dupa ce a primit liniile necesare de la procesul cu id-ul anterior si de la cel cu id-ul urmator, nefiind nevoit sa astepte dupa toate celelalte procese.