Laborator 1

# Analiza cerintei

Programul construit transforma o matrice citita din fisier printr-o formula data, aplicata asupra fiecarui element individual. Executia este realizata atat secvential, cat si paralelizat, utilizand 2 limbaje de programare diferite (Java si C++). Scopul urmarit este de a compara eficienta ca timp a diferitelor metode de executie si realizare, documentate mai jos.

# Proiectare

Ambele proiecte au fost executate cu ajutorul unor scripturi Powershell, facilitand evaluarea mediei ca timp a mai multor cazuri de executie. Parametrii N,M,n,m au fost transmisi din linia de comanda, iar in cadrul codului a fost realizata citirea din fisier a matricilor cu date. Pentru varianta secventiala am realizat o parcurgere pe linii si coloane a intregii matrici f, efectuand calculul cerut dupa anumite conditii, conform specificatiilor. In cazul variantei paralelizate, am folosit un numar variabil p de threaduri care executa operatiile pe un anumit numar de linii (parcurgere pe orizontala), varianta pe care am ales-o datorita reprezentarii matricilor in memorie. La final, programul verifica validitatea calculelor efectuate, comparand cu varianta secventiala si returneaza un calcul al timpului necesar executiei in milisecunde.

# Cazuri de testare

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

# Analiza rezultatelor

In urma rezultatelor obtinute, putem constata o diferenta evidenta intre eficienta in C++ fata de Java, rezultatele fiind considerabil mai bune in toate situatiile evaluate. De asemenea, observam tendinta utilizarii executiei concurente de a fi mai eficienta in anumite situatii, mai ales pe volume de date mari si cu un numar relativ restrans de threaduri. Utilizarea unui numar mare de threaduri devine tot mai ineficienta, rezultat observabil in mod special in Java, unde timpii fluctueaza masiv in testele pe 16 threaduri.

## Rezultate Java

|  |  |  |
| --- | --- | --- |
| Tip Matrice | Nr threads | Timp executie |
| N=10 M=10 n=3 m=3 | 1 | 1.02583 |
| 4 | 1.22936 |
|  |  |  |
| N=1000 M=1000 n=5 m=5 | 1 | 146.79787 |
| 2 | 96.19703 |
| 4 | 74.37376 |
| 8 | 63.82098 |
| 16 | 66.70397 |
|  |  |  |
| N=10 M=10000 n=5 m=5 | 1 | 41.72121 |
| 2 | 45.47922 |
| 4 | 62.63723 |
| 8 | 74.71323 |
| 16 | 68.34673 |
|  |  |  |
| N=10000 M=10 n=5 m=5 | 1 | 28.65365 |
| 2 | 29.83326 |
| 4 | 28.15628 |
| 8 | 45.15832 |
| 16 | 54.29135 |

## Rezultate C++

|  |  |  |  |
| --- | --- | --- | --- |
| Tip Matrice | Tip alocare | Nr threads | Timp executie |
| N=10 M=10 n=3 m=3 | static | 1 | 1.29417 |
| 4 | 1.6023 |
| dinamic | 1 | 1.36991 |
| 4 | 1.71022 |
|  |  |  |  |
| N=1000 M=1000 n=5 m=5 | static | 1 | 175.6237 |
| 2 | 94.39429 |
| 4 | 55.211 |
| 8 | 48.20551 |
| 16 | 44.02512 |
| dinamic | 1 | 175.4795 |
| 2 | 97.04211 |
| 4 | 59.73387 |
| 8 | 52.0899 |
| 16 | 44.94477 |
|  |  |  |  |
| N=10 M=10000 n=5 m=5 | static | 1 | 18.70991 |
| 2 | 10.83072 |
| 4 | 7.82697 |
| 8 | 6.30187 |
| 16 | 6.22725 |
| dinamic | 1 | 18.64684 |
| 2 | 11.43318 |
| 4 | 8.29695 |
| 8 | 6.87958 |
| 16 | 6.77107 |
|  |  |  |  |
| N=10000 M=10 n=5 m=5 | static | 1 | 18.65497 |
| 2 | 10.14053 |
| 4 | 6.99112 |
| 8 | 6.34639 |
| 16 | 5.95449 |
| dinamic | 1 | 20.28226 |
| 2 | 12.27147 |
| 4 | 7.35562 |
| 8 | 6.73294 |
| 16 | 6.17766 |