# C++

|  |  |  |
| --- | --- | --- |
| **Tip matrice** | **Nr threads** | **Timp executie** |
| **N=M=10, n=m=3** | **1** | **4283** |
| **2** | **298544** |
| **N=M=1000, n=m=3** | **1** | **20749440** |
| **2** | **13743327** |
| **4** | **8774705** |
| **8** | **7170077** |
| **16** | **6090241** |
| **N=M=10000, n=m=3** | **1** | **2249872260** |
| **2** | **1280830734** |
| **4** | **628486662** |
| **8** | **475531370** |
| **16** | **381149387** |

# Java

|  |  |  |
| --- | --- | --- |
| **Tip matrice** | **Nr threads** | **Timp executie** |
| **N=M=10, n=m=3** | **1** | **217133** |
| **2** | **3693503** |
| **N=M=1000, n=m=3** | **1** | **42372359** |
| **2** | **36437772** |
| **4** | **40886869** |
| **8** | **51775948** |
| **16** | **101711350** |
| **N=M=10000, n=m=3** | **1** | **2301912029** |
| **2** | **1488741040** |
| **4** | **1060190129** |
| **8** | **731823415** |
| **16** | **607466095** |

Pentru a rezolva problema abordand distributia pe linii si fara a aloca o alt amatrice pentru raspuns, pentru fiecare thread care corespunde unui interval de linii [st, dr], vom avea nevoie sa copiem si liniile st-1, respective dr+1. De asemenea, la fiecare linie curenta vom tine o copie a liniei de mai sus (care initial este st-1, si refolosim acest vector), si vom tine o copie a liniei curente (care va deveni linia “de sus” pentru linia urmatoare), pe care le folosim in calcul; la final vom avea nevoie de ultima linie, care este dr+1 si este deja salvata. In total, vom folosi doar 3 vectori de marime M, deci complexitatea spatiu este O(M). Pentru sincronizarea copierii vectorilor, vom folosi bariere care vor astepta ca toate threadurile sa termine de copiat liniile st-1 si dr+1 corespunzatoare.

Ca si data trecuta, pentru N=M=10, folosirea threadurilor introduce un ovrerhead si merge mai incet decat executia secventiala (inclusive N=M=1000 pentru Java), dar aduce o imbunatatire semnificativa pentru valori mai mari ale matricii. De asemenea, varianta C++ este ~1.5-2x mai rapida decat Java.

Spre deosebire de data trecuta, folosim doar O(M) memorie suplimentara in loc de O(N\*M), ceea ce este un avantaj daca suntem limitati de memorie sau memoria devine un overhead; insa pentru datele noaste de test timpii de executie sunt de ~1.25 ori mai inceti, deoarece avem copierea vectorilor.