**Documentarea performanței**

**Java: (rows)**

**.\runScriptsRows.ps1**

|  |  |  |  |
| --- | --- | --- | --- |
| **Tip matrice** | **Nr threads** | **Timp execuție** | **Comanda** |
| N=M=10  n=m=3 | secvențial | 0,131834 | .\scriptJSequential.ps1 MainSequential "N=M=10 si n=m=3" 10 0 0 |
| 4 | 1,069432 | .\scriptJ.ps1 MainParallel "N=M=10 si n=m=3" 4 10 0 0 |
| N=M=1000  n=m=5 | secvențial | 37,091976 | .\scriptJSequential.ps1 MainSequential "N=M=1000 si n=m=5" 10 1 1 |
| 2 | 25,66447667 | .\scriptJ.ps1 MainParallel "N=M=1000 si n=m=5" 2 10 1 1 |
| 4 | 22,39992667 | .\scriptJ.ps1 MainParallel "N=M=1000 si n=m=5" 4 10 1 1 |
| 8 | 23,55493706 | .\scriptJ.ps1 MainParallel "N=M=1000 si n=m=5" 8 10 1 1 |
| 16 | 27,68397467 | .\scriptJ.ps1 MainParallel "N=M=1000 si n=m=5" 16 10 1 1 |
| N=10 M=10000  n=m=5 | secvențial | 8,417732 | .\scriptJSequential.ps1 MainSequential "N=10 M=10000 si n=m=5" 10 2 1 |
| 2 | 9,534022667 | .\scriptJ.ps1 MainParallel "N=10 M=10000 si n=m=5" 2 10 2 1 |
| 4 | 9,819471333 | .\scriptJ.ps1 MainParallel "N=10 M=10000 si n=m=5" 4 10 2 1 |
| 8 | 10,77548267 | .\scriptJ.ps1 MainParallel "N=10 M=10000 si n=m=5" 8 10 2 1 |
| 16 | 10,376898 | .\scriptJ.ps1 MainParallel "N=10 M=10000 si n=m=5" 16 10 2 1 |
| N=10000 M=10  n=m=5 | secvențial | 13,828236 | .\scriptJSequential.ps1 MainSequential "N=10000 M=10 si n=m=5" 10 3 1 |
| 2 | 14,900804 | .\scriptJ.ps1 MainParallel "N=10000 M=10 si n=m=5" 2 10 3 1 |
| 4 | 14,74691643 | .\scriptJ.ps1 MainParallel "N=10000 M=10 si n=m=5" 4 10 3 1 |
| 8 | 14,899276 | .\scriptJ.ps1 MainParallel "N=10000 M=10 si n=m=5" 8 10 3 1 |
| 16 | 14,17988733 | .\scriptJ.ps1 MainParallel "N=10000 M=10 si n=m=5" 16 10 3 1 |

**Java: (cols)**

**.\runScriptsCols.ps1**

|  |  |  |  |
| --- | --- | --- | --- |
| **Tip matrice** | **Nr threads** | **Timp execuție** | **Comanda** |
| N=M=10  n=m=3 | secvențial | 0,133306 | .\scriptJSequential.ps1 MainSequential "N=M=10 si n=m=3" 10 0 0 |
| 4 | 1,107018 | .\scriptJ.ps1 MainParallelCols "N=M=10 si n=m=3" 4 10 0 0 |
| N=M=1000  n=m=5 | secvențial | 36,415366 | .\scriptJSequential.ps1 MainSequential "N=M=1000 si n=m=5" 10 1 1 |
| 2 | 28,40325467 | .\scriptJ.ps1 MainParallelCols "N=M=1000 si n=m=5" 2 10 1 1 |
| 4 | 24,99196067 | .\scriptJ.ps1 MainParallelCols "N=M=1000 si n=m=5" 4 10 1 1 |
| 8 | 26,25973932 | .\scriptJ.ps1 MainParallelCols "N=M=1000 si n=m=5" 8 10 1 1 |
| 16 | 26,85672143 | .\scriptJ.ps1 MainParallelCols "N=M=1000 si n=m=5" 16 10 1 1 |
| N=10 M=10000  n=m=5 | secvențial | 8,440178 | .\scriptJSequential.ps1 MainSequential "N=10 M=10000 si n=m=5" 10 2 1 |
| 2 | 9,895481429 | .\scriptJ.ps1 MainParallelCols "N=10 M=10000 si n=m=5" 2 10 2 1 |
| 4 | 10,16734601 | .\scriptJ.ps1 MainParallelCols "N=10 M=10000 si n=m=5" 4 10 2 1 |
| 8 | 10,86018461 | .\scriptJ.ps1 MainParallelCols "N=10 M=10000 si n=m=5" 8 10 2 1 |
| 16 | 11,836392 | .\scriptJ.ps1 MainParallelCols "N=10 M=10000 si n=m=5" 16 10 2 1 |
| N=10000 M=10  n=m=5 | secvențial | 13,501814 | .\scriptJSequential.ps1 MainSequential "N=10000 M=10 si n=m=5" 10 3 1 |
| 2 | 15,03643338 | .\scriptJ.ps1 MainParallelCols "N=10000 M=10 si n=m=5" 2 10 3 1 |
| 4 | 14,49590144 | .\scriptJ.ps1 MainParallelCols "N=10000 M=10 si n=m=5" 4 10 3 1 |
| 8 | 15,26085538 | .\scriptJ.ps1 MainParallelCols "N=10000 M=10 si n=m=5" 8 10 3 1 |
| 16 | 15,40864333 | .\scriptJ.ps1 MainParallelCols "N=10000 M=10 si n=m=5" 16 10 3 1 |

**Java: (blocks)**

**.\runScriptsBlocks.ps1**

|  |  |  |  |
| --- | --- | --- | --- |
| **Tip matrice** | **Nr threads** | **Timp execuție** | **Comanda** |
| N=M=10  n=m=3 | secvențial | 0,13397 | .\scriptJSequential.ps1 MainSequential "N=M=10 si n=m=3" 10 0 0 |
| 4 | 1,094092 | .\scriptJ.ps1 MainParallelBlocks "N=M=10 si n=m=3" 4 10 0 0 |
| N=M=1000  n=m=5 | secvențial | 36,537536 | .\scriptJSequential.ps1 MainSequential "N=M=1000 si n=m=5" 10 1 1 |
| 2 | 25,69271333 | .\scriptJ.ps1 MainParallelBlocks "N=M=1000 si n=m=5" 2 10 1 1 |
| 4 | 23,66782864 | .\scriptJ.ps1 MainParallelBlocks "N=M=1000 si n=m=5" 4 10 1 1 |
| 8 | 25,49044727 | .\scriptJ.ps1 MainParallelBlocks "N=M=1000 si n=m=5" 8 10 1 1 |
| 16 | 33,05684471 | .\scriptJ.ps1 MainParallelBlocks "N=M=1000 si n=m=5" 16 10 1 1 |
| N=10 M=10000  n=m=5 | secvențial | 9,3976879 | .\scriptJSequential.ps1 MainSequential "N=10 M=10000 si n=m=5" 10 2 1 |
| 2 | 10,01914733 | .\scriptJ.ps1 MainParallelBlocks "N=10 M=10000 si n=m=5" 2 10 2 1 |
| 4 | 9,80947864 | .\scriptJ.ps1 MainParallelBlocks "N=10 M=10000 si n=m=5" 4 10 2 1 |
| 8 | 10,42636664 | .\scriptJ.ps1 MainParallelBlocks "N=10 M=10000 si n=m=5" 8 10 2 1 |
| 16 | 11,90716935 | .\scriptJ.ps1 MainParallelBlocks "N=10 M=10000 si n=m=5" 16 10 2 1 |
| N=10000 M=10  n=m=5 | secvențial | 13,10753992 | .\scriptJSequential.ps1 MainSequential " N=10000 M=10 si n=m=5" 10 3 1 |
| 2 | 14,75403532 | .\scriptJ.ps1 MainParallelBlocks "N=10000 M=10 si n=m=5" 2 10 3 1 |
| 4 | 14,80187133 | .\scriptJ.ps1 MainParallelBlocks "N=10000 M=10 si n=m=5" 4 10 3 1 |
| 8 | 15,440634 | .\scriptJ.ps1 MainParallelBlocks "N=10000 M=10 si n=m=5" 8 10 3 1 |
| 16 | 14,06513933 | .\scriptJ.ps1 MainParallelBlocks "N=10000 M=10 si n=m=5" 16 10 3 1 |

**C++**

**.\run\_all\_scripts.ps1**

**(rows)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Tip matrice** | **Tip alocare** | **Nr threads** | **Timp execuție** |
| N=M=10  n=m=3 | static | secvențial | 0,016752 |
| 4 | 0,225786 |
| dinamic | secvențial | 0,012092 |
| 4 | 0,230931 |
| N=M=1000  n=m=5 | static | secvențial | 186,4591 |
| 2 | 103,0228 |
| 4 | 54,34603 |
| 8 | 34,42445 |
| 16 | 29,72895 |
| dinamic | secvențial | 217,5608 |
| 2 | 108,1627 |
| 4 | 55,20758 |
| 8 | 35,31183 |
| 16 | 28,50046 |
| N=10 M=10000  n=m=5 | static | secvențial | 19,0736 |
| 2 | 23,774 |
| 4 | 14,34811 |
| 8 | 9,609984 |
| 16 | 5,283537 |
| dinamic | secvențial | 22,6968 |
| 2 | 11,12728 |
| 4 | 6,884261 |
| 8 | 4,917478 |
| 16 | 4,437533 |
| N=10000 M=10  n=m=5 | static | secvențial | 46,01571 |
| 2 | 23,39127 |
| 4 | 11,92637 |
| 8 | 6,405271 |
| 16 | 3,551351 |
| dinamic | secvențial | 21,70927 |
| 2 | 10,99986 |
| 4 | 5,761563 |
| 8 | 3,663123 |
| 16 | 3,414927 |

**C++**

**(cols)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Tip matrice** | **Tip alocare** | **Nr threads** | **Timp execuție** |
| N=M=10  n=m=3 | static | secvențial | 0,017916 |
| 4 | 0,21998 |
| dinamic | secvențial | 0,009694 |
| 4 | 0,224147 |
| N=M=1000  n=m=5 | static | secvențial | 183,963 |
| 2 | 239,885 |
| 4 | 122,7068 |
| 8 | 65,00543 |
| 16 | 30,14636 |
| dinamic | secvențial | 217,385 |
| 2 | 113,3696 |
| 4 | 59,14186 |
| 8 | 37,91746 |
| 16 | 29,65911 |
| N=10 M=10000  n=m=5 | static | secvențial | 18,91419 |
| 2 | 23,38248 |
| 4 | 11,91551 |
| 8 | 6,386422 |
| 16 | 3,553959 |
| dinamic | secvențial | 22,72552 |
| 2 | 11,1436 |
| 4 | 5,762819 |
| 8 | 3,786489 |
| 16 | 3,463007 |
| N=10000 M=10  n=m=5 | static | secvențial | 46,00356 |
| 2 | 23,46401 |
| 4 | 14,18678 |
| 8 | 9,496138 |
| 16 | 5,235542 |
| dinamic | secvențial | 21,82295 |
| 2 | 10,99053 |
| 4 | 6,873623 |
| 8 | 4,995041 |
| 16 | 4,505282 |

**C++**

**(blocks)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Tip matrice** | **Tip alocare** | **Nr threads** | **Timp execuție** |
| N=M=10  n=m=3 | static | secvențial | 0,01671 |
| 4 | 0,243277 |
| dinamic | secvențial | 0,009636 |
| 4 | 0,22958 |
| N=M=1000  n=m=5 | static | secvențial | 183,2617 |
| 2 | 237,4935 |
| 4 | 120,2154 |
| 8 | 62,98938 |
| 16 | 27,09368 |
| dinamic | secvențial | 216,4962 |
| 2 | 109,531 |
| 4 | 56,20619 |
| 8 | 34,89612 |
| 16 | 28,73472 |
| N=10 M=10000  n=m=5 | static | secvențial | 18,95628 |
| 2 | 24,01105 |
| 4 | 12,23079 |
| 8 | 6,539817 |
| 16 | 3,57189 |
| dinamic | secvențial | 22,93709 |
| 2 | 10,93837 |
| 4 | 5,784008 |
| 8 | 3,822385 |
| 16 | 3,655637 |
| N=10000 M=10  n=m=5 | static | secvențial | 45,85972 |
| 2 | 23,45278 |
| 4 | 11,94387 |
| 8 | 6,392414 |
| 16 | 3,558542 |
| dinamic | secvențial | 22,03958 |
| 2 | 11,02744 |
| 4 | 5,821875 |
| 8 | 3,800155 |
| 16 | 3,433954 |

**Analiza**

* Performanța pentru fiecare caz
  + Java
    - Doar pentru cazul 1000x1000 timpul pentru secvențial este mai mare decât pentru paralel. În celelalte, varianta secvențială este un pic mai rapidă.
  + C++
    - Aici, cazul 1000x1000 static secvențial obține un timp mai bun decât paralel. În celelalte cazuri, în general timpul secvențial este aproape dublu față de cel paralel, că e vorba de static ori dinamic.
* Timpii din Java vs. timpii din C++
  + Pentru variantele secvențiale, C++ obține timpi foarte mari (cu mult)
  + Între cele din Java nu sunt diferențe enorme de perfomanță de la un număr de threaduri la altul, însă în C++ se îmbunătățește timpul, în general, cu cât crește numărul de threaduri (aproape se înjumătățește la fiecare dublare a numărului de threaduri). Per total, între Java și C++, cazul 1000x1000 este favorabil pentru Java. În schimb, celelalte obțin timpi mai buni în C++, mai ales când ajungem la 16 threaduri.
* C++ static vs. dinamic
  + În cazul implementării statice, diferențele de la un număr de threaduri la altul aduc aproape îmbunătățiri de înjumătățire a timpului (la 8 și 16 threaduri sunt apropiate valorile performanței). În cazul dinamic, nu e chiar așa bruscă trecerea, cum se poate vedea pe diagrame. Totuși, în mare, timpii scoși de implementările dinamice par a fi mai buni decât cei dați de implementările statice (cu excepția implementării secvențiale).

**Implementare**

* Pe linii
  + Se distribuie thread-urilor linii N/P și, în cazul în care există rest, primele N%P thread-uri vor primi câte o linie în plus (mai precis, se calculează liniile de start și end pentru fiecare thread). Instrucțiunile de ciclare pe linii și coloane merg cu liniile de la start la end, iar coloanele de la 0 la M.
* Pe coloane
  + Se distribuie thread-urilor coloane M/P și, în cazul în care există rest, primele M%P thread-uri vor primi câte o coloană în plus (mai precis, se calculează coloanele de start și end pentru fiecare thread). Instrucțiunile de ciclare pe coloane și linii merg cu coloanele de la start la end, iar liniile de la 0 la N.
* Blocuri
  + Se consideră (ca idee) că matricea este un vector de dimensiune N\*M. Aici thread-urile primesc N\*M/P elemente, +1 în cazul primelor N\*M%P thread-uri. Pentru a afla indicii de linie și coloană se face index/M, respectiv index%M.

**Diagrama de clase din Java**

O imagine care conține text, număr, diagramă, captură de ecran

Descriere generată automat