Laborator 1

Java:

|  |  |  |
| --- | --- | --- |
| **Tip matrice** | **Nr threads** | **Timp executie** |
| N=M=10  n=m=3 | secvential | 26820 ns |
| 4 - vertical | 377290 ns |
| 4 - orizontal | 421510 ns |
| N=M=1000  n=m=5 | secvential | 23567430 ns |
| 2 - vertical | 13020310 ns |
| 2 - orizontal | 13258270 ns |
| 4 - vertical | 6383960 ns |
| 4 - orizontal | 7317740 ns |
| 8 - vertical | 6325890 ns |
| 8 - orizontal | 7543560 ns |
| 16 - vertical | 6562480 ns |
| 16 - orizontal | 5548630 ns |
| N=10 M=10000 n=m=5 | secvential | 3937810 ns |
| 2 - vertical | 3337610 ns |
|  | 2 - orizontal | 5485360 ns |
|  | 4 - vertical | 1057760 ns |
|  | 4 - orizontal | 1214280 ns |
|  | 8 - vertical | 1161700 ns |
|  | 8 - orizontal | 1324010 ns |
|  | 16 - vertical | 1541530 ns |
| 16 - orizontal | 1676480ns |
| N=10000 M=10 n=m=5 | secvential | 2841760 ns |
|  | 2 - vertical | 3920940 ns |
| 2 - orizontal | 3475930 ns |
| 4 - vertical | 1396450 ns |
| 4 - orizontal | 1461910 ns |
| 8 - vertical | 1312330 ns |
| 8 - orizontal | 1550670 ns |
| 16 - vertical | 2093860 ns |
| 16 - orizontal | 2005000 ns |
| N=10000 M=10000  n=m=5 | secvential | 2157853680 ns |
|  | 2 - vertical | 1159725630 ns |
|  | 2 - orizontal | 1159725630 ns |
|  | 4 - vertical | 510289200 ns |
|  | 4 - orizontal | 510143470 ns |
|  | 8 - vertical | 458518700 ns |
|  | 8 - orizontal | 414504780 ns |
|  | 16 - vertical | 306501000 ns |
|  | 16 - orizontal | 308059750 ns |

C++ static:

|  |  |  |
| --- | --- | --- |
| **Tip matrice** | **Nr threads** | **Timp executie** |
| N=M=10  n=m=3 | secvential | 2210 ns |
| 4 - vertical | 286130 ns |
| 4 - orizontal | 233070 ns |
| N=M=1000  n=m=5 | secvential | 23567430 ns |
| 2 - vertical | 13020300 ns |
| 2 - orizontal | 13258270 ns |
| 4 - vertical | 6383960 ns |
| 4 - orizontal | 7317740 ns |
| 8 - vertical | 6325890 ns |
| 8 - orizontal | 7543560 ns |
| 16 - vertical | 6562480 ns |
| 16 - orizontal | 5548630 ns |
| N=10 M=10000 n=m=5 | secvential | 3937810 ns |
| 2 - vertical | 3337610 ns |
|  | 2 - orizontal | 5485360 ns |
|  | 4 - vertical | 1057760 ns |
|  | 4 - orizontal | 1214280 ns |
|  | 8 - vertical | 1161700 ns |
|  | 8 - orizontal | 1324010 ns |
|  | 16 - vertical | 1541530 ns |
| 16 - orizontal | 1676480 ns |
| N=10000 M=10 n=m=5 | secvential | 2841760 ns |
|  | 2 - vertical | 3920940 ns |
| 2 - orizontal | 3475930 ns |
| 4 - vertical | 1396450 ns |
| 4 - orizontal | 1461910 ns |
| 8 - vertical | 1312330 ns |
| 8 - orizontal | 1550670 ns |
| 16 - vertical | 2093860 ns |
| 16 - orizontal | 2005000 ns |
| N=10000 M=10000  n=m=5 | secvential | 2157853680 ns |
|  | 2 - vertical | 1159725630 ns |
|  | 2 - orizontal | 1159725630 ns |
|  | 4 - vertical | 510289200 ns |
|  | 4 - orizontal | 510143470ns |
|  | 8 - vertical | 458518700 ns |
|  | 8 - orizontal | 414504780 ns |
|  | 16 - vertical | 306501000 ns |
|  | 16 - orizontal | 308059750 ns |

C++ dinamic:

|  |  |  |
| --- | --- | --- |
| **Tip matrice** | **Nr threads** | **Timp executie** |
| N=M=10  n=m=3 | secvential | 2160 ns |
| 4 - vertical | 412240 ns |
| 4 - orizontal | 265960 ns |
| N=M=1000  n=m=5 | secvential | 47370710 ns |
| 2 - vertical | 23938230 ns |
| 2 - orizontal | 24733330 ns |
| 4 - vertical | 12498620 ns |
| 4 - orizontal | 12842900 ns |
| 8 - vertical | 9574520 ns |
| 8 - orizontal | 9539770 ns |
| 16 - vertical | 7741400 ns |
| 16 - orizontal | 8419600 ns |
| N=10 M=10000 n=m=5 | secvential | 4781230 ns |
| 2 - vertical | 2533680 ns |
|  | 2 - orizontal | 2951730 ns |
|  | 4 - vertical | 1571670 ns |
|  | 4 - orizontal | 2208580 ns |
|  | 8 - vertical | 1565380 ns |
|  | 8 - orizontal | 2215680 ns |
|  | 16 - vertical | 1486590 ns |
| 16 - orizontal | 5641930 ns |
| N=10000 M=10 n=m=5 | secvential | 4826510 ns |
|  | 2 - vertical | 2763640 ns |
| 2 - orizontal | 2743430 ns |
| 4 - vertical | 2391260 ns |
| 4 - orizontal | 1727500 ns |
| 8 - vertical | 2273550 ns |
| 8 - orizontal | 1657810 ns |
| 16 - vertical | 5467580 ns |
| 16 - orizontal | 1631590 ns |
| N=10000 M=10000  n=m=5 | secvential | 4928283460 ns |
|  | 2 - vertical | 2470337550 ns |
|  | 2 - orizontal | 2480096040 ns |
|  | 4 - vertical | 2966692120 ns |
|  | 4 - orizontal | 3023509690 ns |
|  | 8 - vertical | 1484376560 ns |
|  | 8 - orizontal | 1496362890 ns |
|  | 16 - vertical | 1051300630 ns |
|  | 16 - orizontal | 1176577760 ns |

Comparatia secvential versus paralel

* Java:
* Pentru dimensiuni mici ale matricii (N=M=10, N=M=1000) si pentru dimensiuni mari (N=M=10000), timpii de executie secventiali si cei paraleli sunt aproximativ egali.
* Atat metoda de paralelizare verticala, cat si cea orizontala, indiferent de numarul de thread-uri, nu aduc imbunatatiri semnificative.
* Acest lucru arata ca overhead-ul thread-urilor in Java este relativ mare, iar paralelizarea nu produce un speedup vizibil in cazul matricilor mici si medii.
* C++:
* Atat pentru matrici mici (N=M=10, N=M=1000) cat si pentru matrici mari (N=M=10000), timpii paraleli scad clar fata de secvential.
* Creste eficienta pe masura ce numarul de thread-uri creste.
* Exemplu: N=M=10000, model dinamic, secvential ≈ 6.5 miliarde ns, iar folosind 16 thread-uri timpul scade la ≈ 1 miliard ns → speedup de aproximativ 6×.
* Modelul static este putin mai rapid decat cel dinamic, dar diferenta nu este semnificativa.

Comparatia variantelor paralele

* Atat in Java, cat si in C++, paralelizarea pe verticala versus orizontal produce timpi similari pentru matrici patratoase mari.
* Pentru matrici de ex: N=10, M=10000 sau N=10000, M=10, paralelizarea pe directia lunga poate fi usor mai eficienta, dar diferentele depind de impartirea sarcinilor si de overhead-ul thread-urilor.

Comparatia Java versus C++

* + Implementarea C++ este mai eficienta decat cea Java in toate cazurile.
  + Exemplu: N=10, M=10000, secvential: Java ≈ 10 milioane ns, C++ ≈ 5 milioane ns → C++ de 2× mai rapid.
  + N=M=10000, 16 thread-uri: C++ ≈ 1 miliard ns, Java ≈ 4 miliarde ns → C++ de 4× mai rapid.
  + Motivul principal: C++ ruleaza nativ, thread-urile sunt mai eficiente si nu exista overhead-ul JVM.

Comparatia C++ static versus dinamic

* + Modelul static este usor mai rapid decat cel dinamic, mai ales pentru matrici mari, datorita accesului mai rapid in memorie (stiva/global vs heap).
  + Trendul de paralelizare este identic: cresterea numarului de thread-uri reduce timpul de executie.
  + Diferentele nu sunt dramatice, dar pentru valori foarte mari de N si M, staticul poate fi mai stabil si mai rapid.