**C++**

**Gestionarea Bordurii**

Pentru a aplica convoluția, matricea de intrare trebuie să fie extinsă cu o bordură (padding) pentru a permite kernel-ului să fie aplicat pe toate elementele matricei originale. Funcția padMatrix din clasa Utils adaugă această bordură.

**Alocarea Rândurilor Thread-urilor**

Pentru a distribui sarcina între mai multe thread-uri, matricea de intrare este împărțită în rânduri, iar fiecare thread procesează un subset de rânduri. Numărul de rânduri alocate fiecărui thread este calculat .

Fiecare thread primește rowsPerThread rânduri, iar rândurile rămase (extraRows) sunt distribuite uniform între primele thread-uri.

**Utilizarea fara matrice auxiliara**

În metoda paralel\_h, vectorii auxiliari sunt utilizați pentru a gestiona condițiile limită și a stoca rezultatele intermediare. Iată o scurtă explicație a modului în care sunt utilizați acești vectori::

* last\_modified: Acest vector stochează valorile rândului de deasupra rândului curent procesat. Este inițializat cu zerouri și actualizat pe măsură ce procesarea trece la rândul următor.
* boundary\_down: Acest vector stochează valorile rândului de sub rândul curent procesat. De asemenea, este inițializat cu zerouri și actualizat în consecință.
* curent: Acest vector stochează rezultatele intermediare pentru rândul curent procesat. Este inițializat cu zerouri și actualizat cu rezultatele convoluției.

**Utilizarea Barierei**

O barieră (std::barrier) este utilizată pentru a sincroniza thread-urile, asigurându-se că toate thread-urile au terminat procesarea înainte de a continua. Fiecare thread apelează barrier.arrive\_and\_wait() după ce își termină sarcina: barrier.arrive\_and\_wait();

**Java**

**Gestionarea Bordurii**

Pentru a aplica convoluția, matricea de intrare trebuie să fie extinsă cu o bordură (padding) pentru a permite kernel-ului să fie aplicat pe toate elementele matricei originale. Funcția padMatrix din clasa Utils adaugă această bordură.

**Alocarea Rândurilor Thread-urilor**

Pentru a distribui sarcina între mai multe thread-uri, matricea de intrare este împărțită în rânduri, iar fiecare thread procesează un subset de rânduri. Numărul de rânduri alocate fiecărui thread este calculat .

Fiecare thread primește rowsPerThread rânduri, iar rândurile rămase (extraRows) sunt distribuite uniform între primele thread-uri.

**Utilizarea fara matrice auxiliara**

* În metoda paralel\_h, vectorii auxiliari sunt utilizați pentru a gestiona condițiile limită și a stoca rezultatele intermediare. Iată o scurtă explicație a modului în care sunt utilizați acești vectori::
* last\_modified: Acest vector stochează valorile rândului de deasupra rândului curent procesat. Este inițializat cu zerouri și actualizat pe măsură ce procesarea trece la rândul următor.
* boundary\_down: Acest vector stochează valorile rândului de sub rândul curent procesat. De asemenea, este inițializat cu zerouri și actualizat în consecință.
* curent: Acest vector stochează rezultatele intermediare pentru rândul curent procesat. Este inițializat cu zerouri și actualizat cu rezultatele convoluției.

**Utilizarea Barierei**

O barieră (CyclicBarrier) este utilizată pentru a sincroniza thread-urile, asigurându-se că toate thread-urile au terminat procesarea înainte de a continua. Fiecare thread apelează barrier.await() după ce își termină sarcina:

*barrier.await();*

Testare pe rulari multiple(10). Pentru acest pas am utilizat scriptJ.ps1 si scriptC.ps1 . Timpul a fost masurat de dinaintea inceperii executiei pana la finalul acesteia.

Cea mai eficineta este implementarea in C++. Per total, media timpilor de rulare este mai buna la acest laborator, fata de laboratorul anterior.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Dimensiune Matrice | Limbaj programare | Type | Nr threads | Medie dupa 10 rulari |
| n=m=10 si k=3 | Java | Secvential | - | 0.058 |
| Linii | 2 | 2.4127 |
| C++ | Secvential |  | 0.0067 |
| Linii | 2 | 0.56 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Dimensiune Matrice | Limbaj programare | Type | Nr threads | Medie dupa 10 rulari |
| n=m=1000 si k =3 | Java | Secvential | - | 35.3011 |
| Linii | 2 | 20.8002 |
| 4 | 23.7895 |
| 8 | 27.2552 |
| 16 | 43.0948 |
| C++ | Secvential | - | 5.3834 |
| Linii | 2 | 4.8203 |
| 4 | 3.8315 |
| 8 | 4.4768 |
| 16 | 4.3672 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Dimensiune Matrice | Limbaj programare | Type | Nr threads | Medie dupa 10 rulari |
| n=m=10000 si k =3 | Java | Secvential | - | 1205.4369 |
| Linii | 2 | 780.2486 |
| 4 | 550.545 |
| 8 | 407.8988 |
| 16 | 348.4726 |
| C++ | Secvential | - | 566.15 |
| Linii | 2 | 428.39 |
| 4 | 319.043 |
| 8 | 322.329 |
| 16 | 271.945 |