**Documentarea performanței**

|  |  |  |  |
| --- | --- | --- | --- |
| **N = M** | **K** | **P** | **Timp execuție** |
| 10 | 3 | secvențial | 0.00983 |
| 4 | 94.55717 |
| 8 | 75.52364 |
| 16 | 73.81904 |
| 32 | 74.33906 |
| 1000 | 3 | secvențial | 82.91577 |
| 4 | 1666.60194 |
| 8 | 1668.45703 |
| 16 | 1682.41924 |
| 32 | 1708.3898 |
| 10000 | 3 | secvențial | 8222.87495 |
| 4 | 157684.08256 |
| 8 | 157585.72386 |
| 16 | 158471.56223 |
| 32 | 157536.16358 |

**Implementare**

1. **Alocarea memoriei pe dispozitiv CUDA:**
   * Folosirea funcției **cudaMalloc** pentru a aloca spațiu pe dispozitiv pentru matricea de intrare (**deviceInputMatrix**), matricea de convoluție (**deviceConvolutionMatrix**) și matricea de ieșire (**deviceOutputMatrix**).
2. **Transferul datelor de la gazdă la dispozitiv:**
   * Utilizarea funcției **cudaMemcpy** pentru a transfera datele din matricele de pe gazdă în cele alocate pe dispozitiv (**cudaMemcpyHostToDevice**).
3. **Configurarea dimensiunilor blocului și grilei:**
   * Stabilirea dimensiunilor blocului și grilei în funcție de **P** și dimensiunile matricii, folosind **dim3**.
4. **Lansarea kernelului CUDA:**
   * Implementarea unui kernel CUDA (**matrixMultiply**) care să efectueze operația de convoluție pentru fiecare element al matricei de ieșire.
5. **Transferul rezultatelor de la dispozitiv la gazdă:**
   * Utilizarea funcției **cudaMemcpy** pentru a transfera matricea de ieșire de la dispozitiv la gazdă (**cudaMemcpyDeviceToHost**).
6. **Dealocarea memoriei pe dispozitivul CUDA:**
   * Utilizarea funcțiilor **cudaFree** pentru a elibera memoria alocată pe dispozitiv.
7. **Funcția de apel pentru CUDA:**
   * Implementarea funcției **startCUDA** care să cuprindă toți pașii de la alocarea memoriei până la dealocarea acesteia, inclusiv lansarea kernelului CUDA.