|  |  |
| --- | --- |
| **N=100** | |
| **CUDA** | **528531** |
| **CUDA Optimised** | **1406** |
| **CPU** | **3198** |
| **N=1000** | |
| **CUDA** | **655585** |
| **CUDA Optimised** | **15714** |
| **CPU** | **2554695** |

Pentru varianta CUDA, impartim matricea in block-uri de dimensiune MxM (am folosit M=10). Fiecare thread va calcula raspunsul pentru o singura valoare a matricei rezultat, citind datele din memoria globala.

Pentru varianta CUDA Optimised, ne folosim de memoria shared care este mult mai rapida.

A screenshot of a computer

Description automatically generated

Pentru toate threadurile dintr-un block, vom refolosi cate un block din matricea A si matricea B pe care il vom copia in shared memory (o singura data). Matricea se va imparti astfel in mai multe blockuri pe orizontala pentru A si mai multe pe verticala pentru B. Folosindu-ne de ce avem in memoria shared la un anumit moment (un block A si un block corespunzator B), putem calcula un raspuns partial la fiecare pas. Mergem pas cu pas, ne asiguram ca am copiat de ce avem nevoie in memoria shared folosind \_\_syncthreads iar apoi putem calcula raspunsurile partiale, care insumate ne vor da raspunsul final.