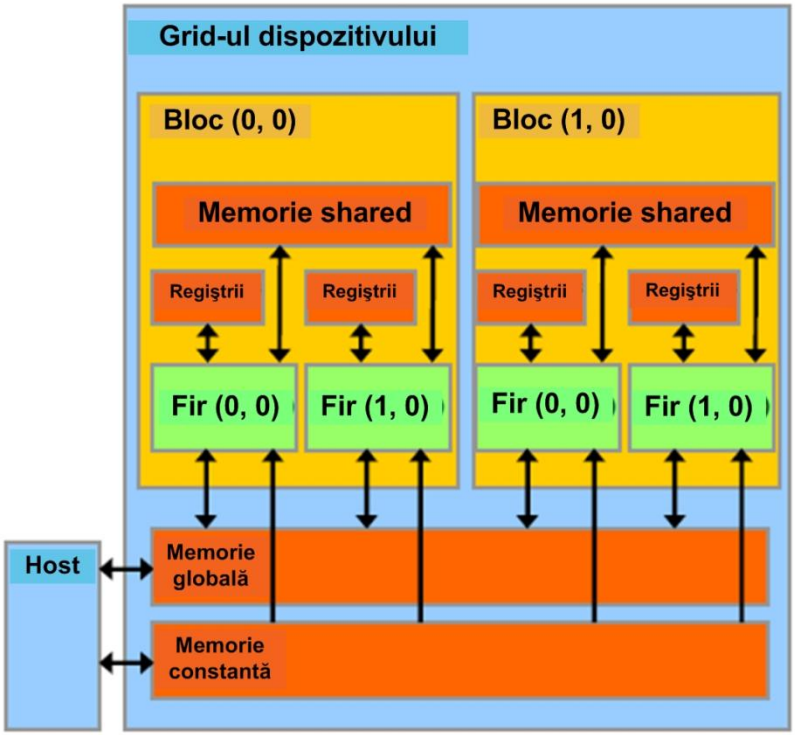
Introducere în CUDA
Continuare

Cosmin – Ioan Niță

Tipuri de memorii CUDA

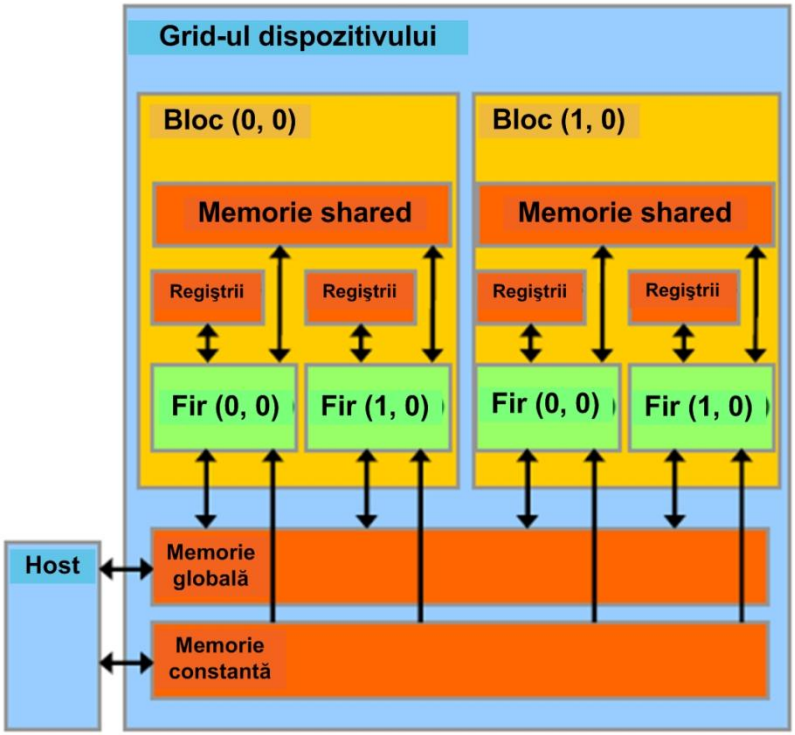
- Memorie globală
- Memorie partajată (shared memory)
- Regiștrii

- Caracteristici:
  - Vizibilitate
  - Perioadă de viață
  - Latență



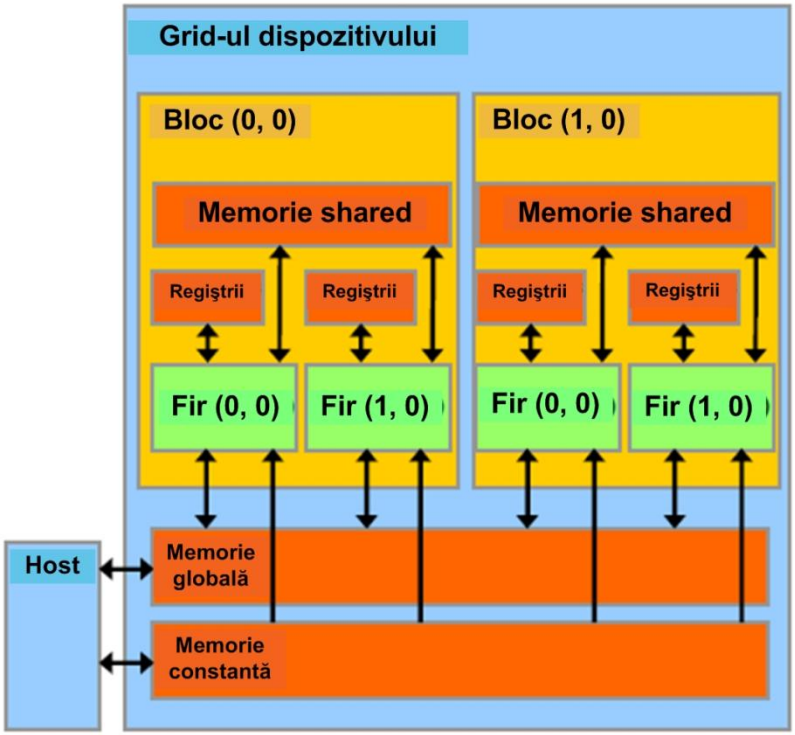
Tipuri de memorii CUDA

- Memorie globală
  - Se alocă și eliberează doar de pe CPU (*cudaMalloc, cudaFree)*
  - Poate fi accesată atât de CPU cât și de GPU (*cudaMemset, cudaMemcpy)*.
  - La nivel de GPU este vizibilă orcărui thread.
  - Accesul este foarte lent.



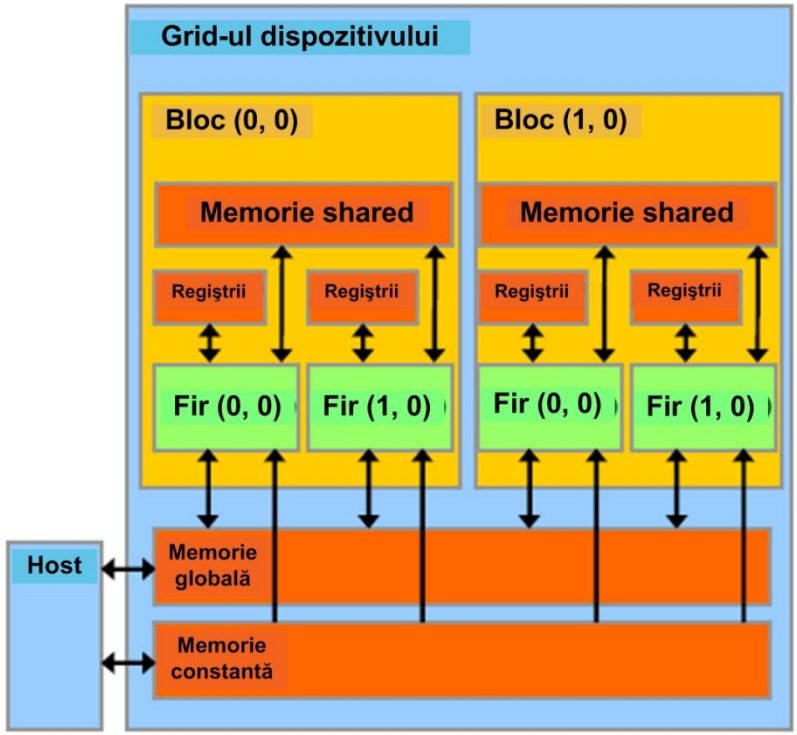
Tipuri de memorii CUDA

- Memorie partajată (shared memory)
  - Vizibilă doar de pe GPU la nivelul unui bloc.
  - Se alocă în interiorul unui kernel. Ex: *__shared__ float f[128];*
  - Accesul este foarte rapid.
  - Limitată, mult mai mică decât memoria globală



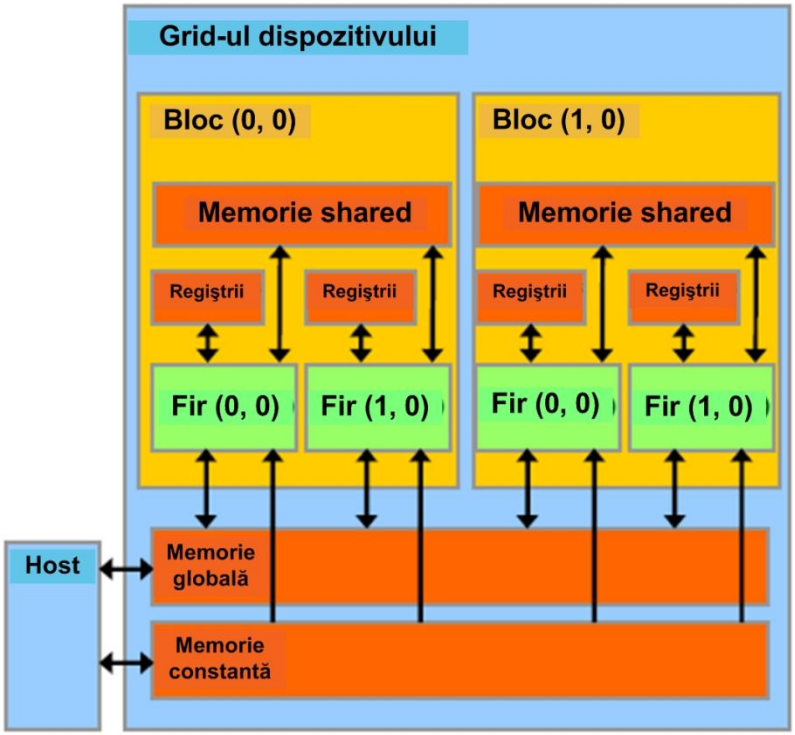
Tipuri de memorii CUDA

- Regiștrii
  - Vizibili doar la nivel de thread.
  - Variabilele declarate local într-un thread vor fi automat plasate în regiștrii.
  - Nu pot fi declarați ca un tablou.
  - Accesul este foarte rapid.



Tipuri de memorii CUDA

- Durata de viață
  - Regiștrii: kernel
  - Mem. Partajată: kernel
  - Mem. Globală: Aplicație



Tipuri de memorii CUDA

- Utilizare
  - Regiștrii:

```
__global__ void kernel()
{
   //Registrii
   float f1,f2,f3;
}
```

Tipuri de memorii CUDA

- Utilizare
  - Memoria partajata:

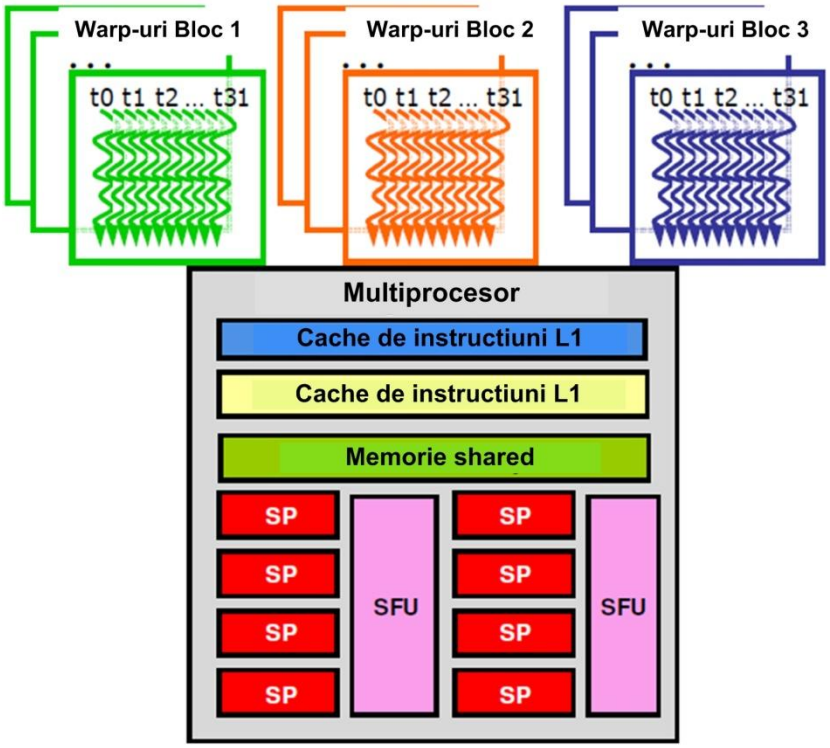
```
__global__ void kernel()
{
   //Shared memory
   __shared__ float tmp[128];
}
```

Factori ce limitează paralelismul

- Dacă se depășește numărul maxim de regiștrii aceștia vor vi alocați în memoria globală.
- Dacă se depășește cantitatea de memorie shared se reduce paralelismul
- DeviceQuery
- Opținuea --ptxas-options=-v

Optimizarea accesului la memoria globală

- Accesul la memoria globală – factor ce limitează viteza de execuție.
- Mascarea latenței
- Cand un warp asteaptă date de la mem. Globală, GPU-ul porneste execuția altui warp.
- Citirea din mem. Globală se poate face în paralel pentru jumătate de warp dacă se accesează adrese de memorie consecutive.



SFU – special function unit = unitate de calcul a operațiilor speciale

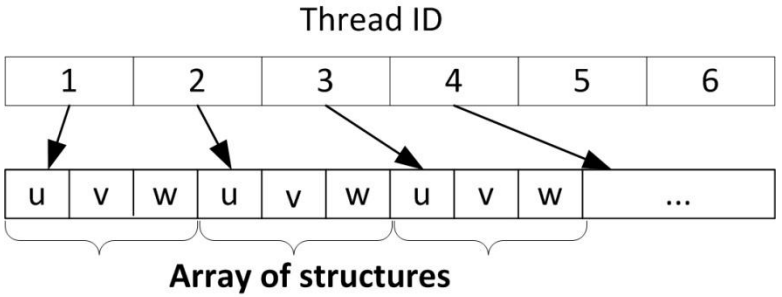
Optimizarea accesului la memoria globală

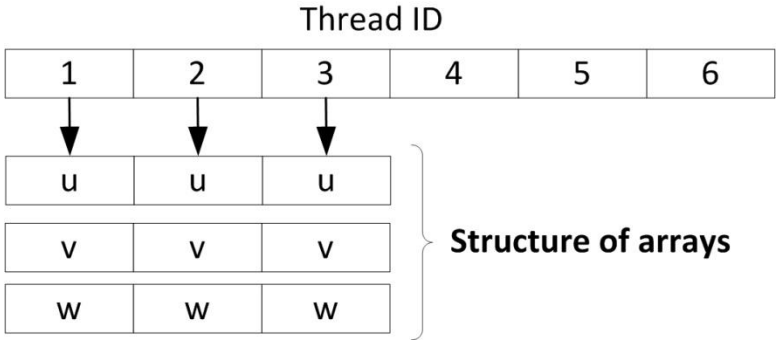
- Exemplu:
  - Transformarea unei mulțimi de puncte 3D.
  - Fiecare thread va executa o transformare.

$$p_i \leftarrow T \cdot p_i$$

$$p_i = [u_i, v, w_i]^T$$

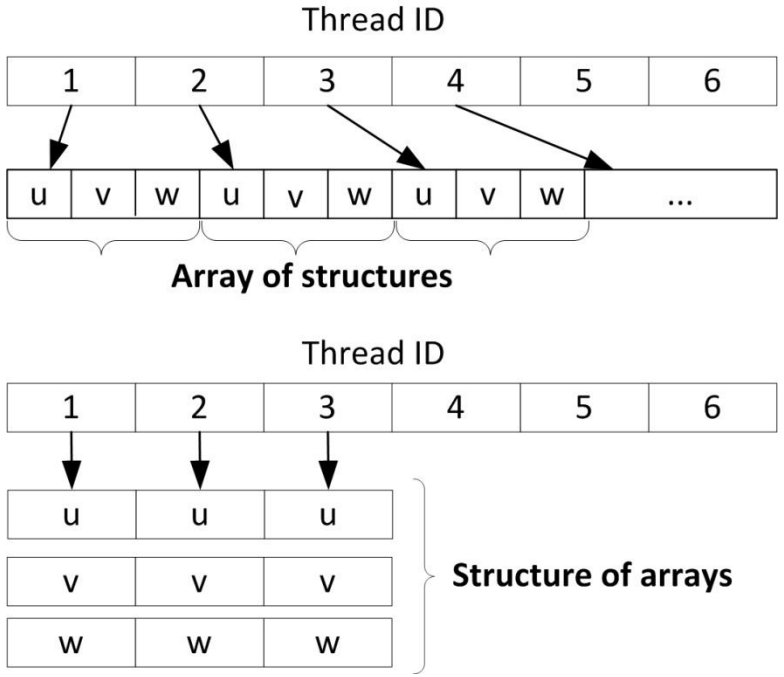
$$T = \begin{bmatrix} t_{xx} & t_{xy} & t_{xz} \\ t_{yx} & t_{yy} & t_{yz} \\ t_{zx} & t_{zy} & t_{zz} \end{bmatrix}$$





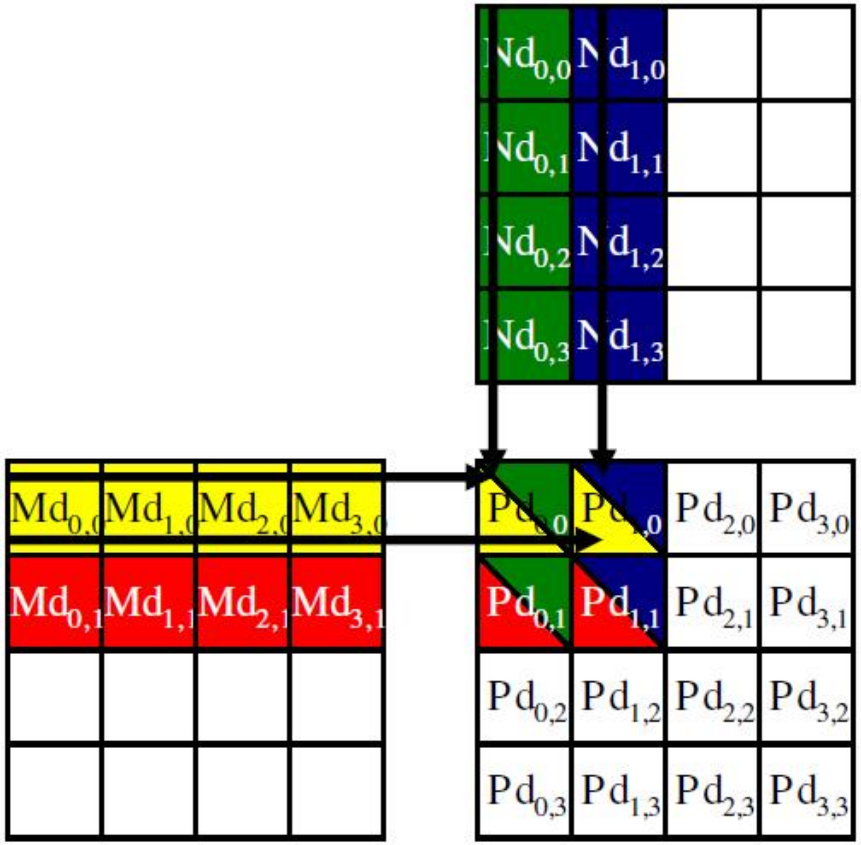
Optimizarea accesului la memoria globală

- Exemplu:
  - Înmulțirea a două matrici.



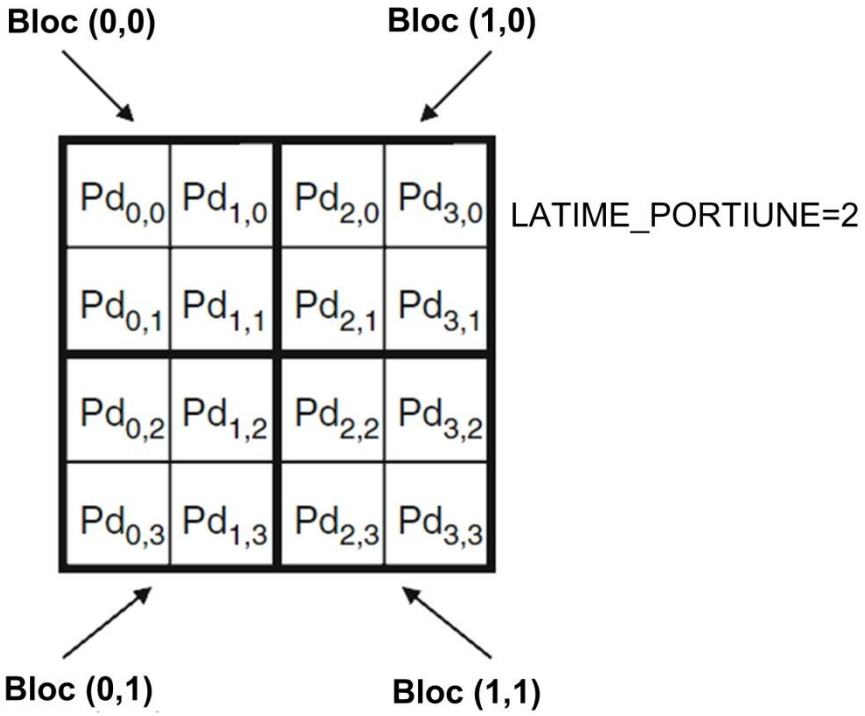
Optimizarea unei aplicații CUDA folosind memoria partajata

- Exemplu pentru înmulțirea a două matrici
  - Accesări redundante ale memoriei globale.

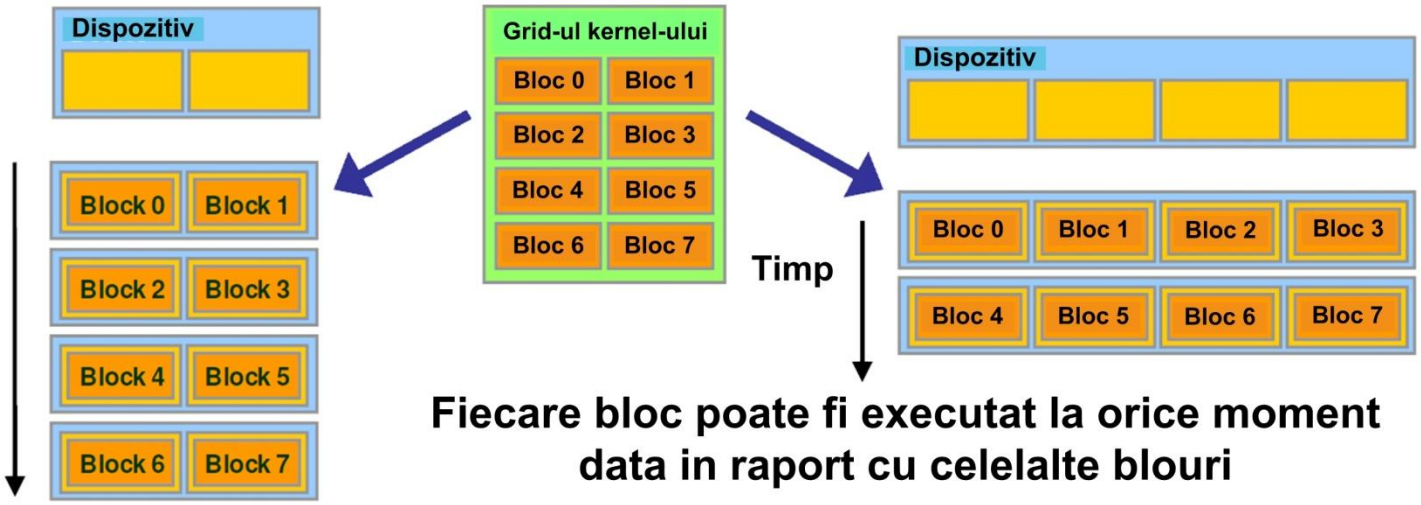


Optimizarea unei aplicații CUDA folosind memoria partajata

- Exemplu pentru înmulțirea a două matrici
  - Accesări redundante ale memoriei globale.
  - Produse partiale
  - Fiecare thread va copia valori din mem. Gobală în memoria shared.



Optimizarea unei aplicații CUDA folosind memoria partajata



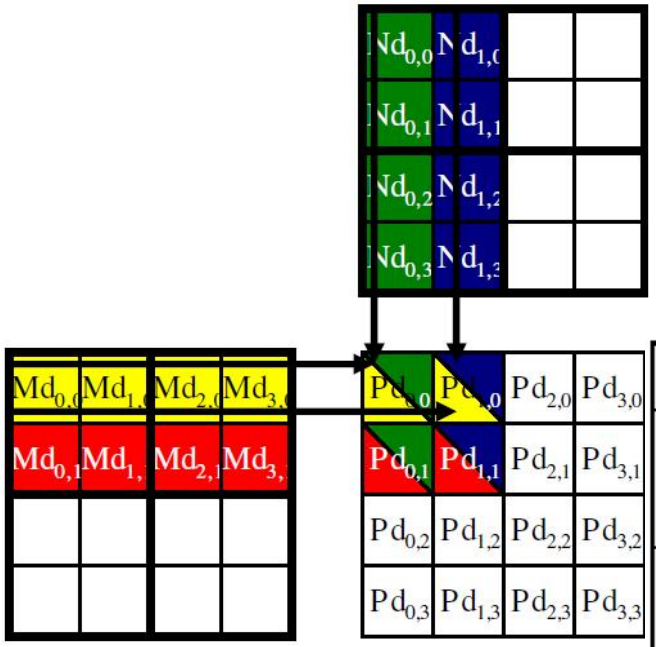
Optimizarea unei aplicații CUDA folosind memoria partajata

- Bariera de sincronizare a thread-urilor dintr-un bloc.
  - __synchtreads();

Optimizarea unei aplicații CUDA folosind memoria partajata

- Bariera de sincronizare a thread-urilor dintr-un bloc.
  - __synchtreads();
  - Deadlock

Optimizarea unei aplicații CUDA folosind memoria partajata



	Faza 1			Faza 2		
$T_{0,0}$	$\mathbf{Md_{0,0}}$ ↓ $Mds_{0,0}$	$\mathbf{Nd_{0,0}}$ ↓ $Nds_{0,0}$	$Pval_{0,0}$+= $Mds_{0,0}$*$Nds_{0,0}$ + $Mds_{1,0}$*$Nds_{0,1}$	$\mathbf{Md_{2,0}}$ ↓ $Mds_{0,0}$	$\mathbf{Nd_{0,2}}$ ↓ $Nds_{0,0}$	$Pval_{0,0}$+= $Mds_{0,0}$*$Nds_{0,0}$ + $Mds_{1,0}$*$Nds_{0,1}$
$T_{1,0}$	$\mathbf{Md_{1,0}}$ ↓ $Mds_{1,0}$	$\mathbf{Nd_{1,0}}$ ↓ $Nds_{1,0}$	$Pval_{1,0}$+= $Mds_{0,0}$*$Nds_{1,0}$ + $Mds_{1,0}$*$Nds_{1,1}$	$\mathbf{Md_{3,0}}$ ↓ $Mds_{1,0}$	$\mathbf{Nd_{1,2}}$ ↓ $Nds_{1,0}$	$Pval_{1,0}$+= $Mds_{0,0}$*$Nds_{1,0}$ + $Mds_{1,0}$*$Nds_{1,1}$
$T_{0,1}$	$\mathbf{Md_{0,1}}$ ↓ $Mds_{0,1}$	$\mathbf{Nd_{0,1}}$ ↓ $Nds_{0,1}$	$Pval_{0,1}$+= $Mds_{0,1}$*$Nds_{0,0}$ + $Mds_{1,1}$*$Nds_{0,1}$	$\mathbf{Md_{2,1}}$ ↓ $Mds_{0,1}$	$\mathbf{Nd_{0,3}}$ ↓ $Nds_{0,1}$	$Pval_{0,1}$+= $Mds_{0,1}$*$Nds_{0,0}$ + $Mds_{1,1}$*$Nds_{0,1}$
$T_{1,1}$	$\mathbf{Md_{1,1}}$ ↓ $Mds_{1,1}$	$\mathbf{Nd_{1,1}}$ ↓ $Nds_{1,1}$	$Pval_{1,1}$+= $Mds_{0,1}$*$Nds_{1,0}$ + $Mds_{1,1}$*$Nds_{1,1}$	$\mathbf{Md_{3,1}}$ ↓ $Mds_{1,1}$	$\mathbf{Nd_{1,3}}$ ↓ $Nds_{1,1}$	$Pval_{1,1}$+= $Mds_{0,1}$*$Nds_{1,0}$ + $Mds_{1,1}$*$Nds_{1,1}$

Timp →