POLYTECH
PARIS-SACLAY

GP-GPU

TD3: Dense matrix product on GPU using shared memory

Stéphane Vialle
Stephane.Vialle@centralesupelec.fr

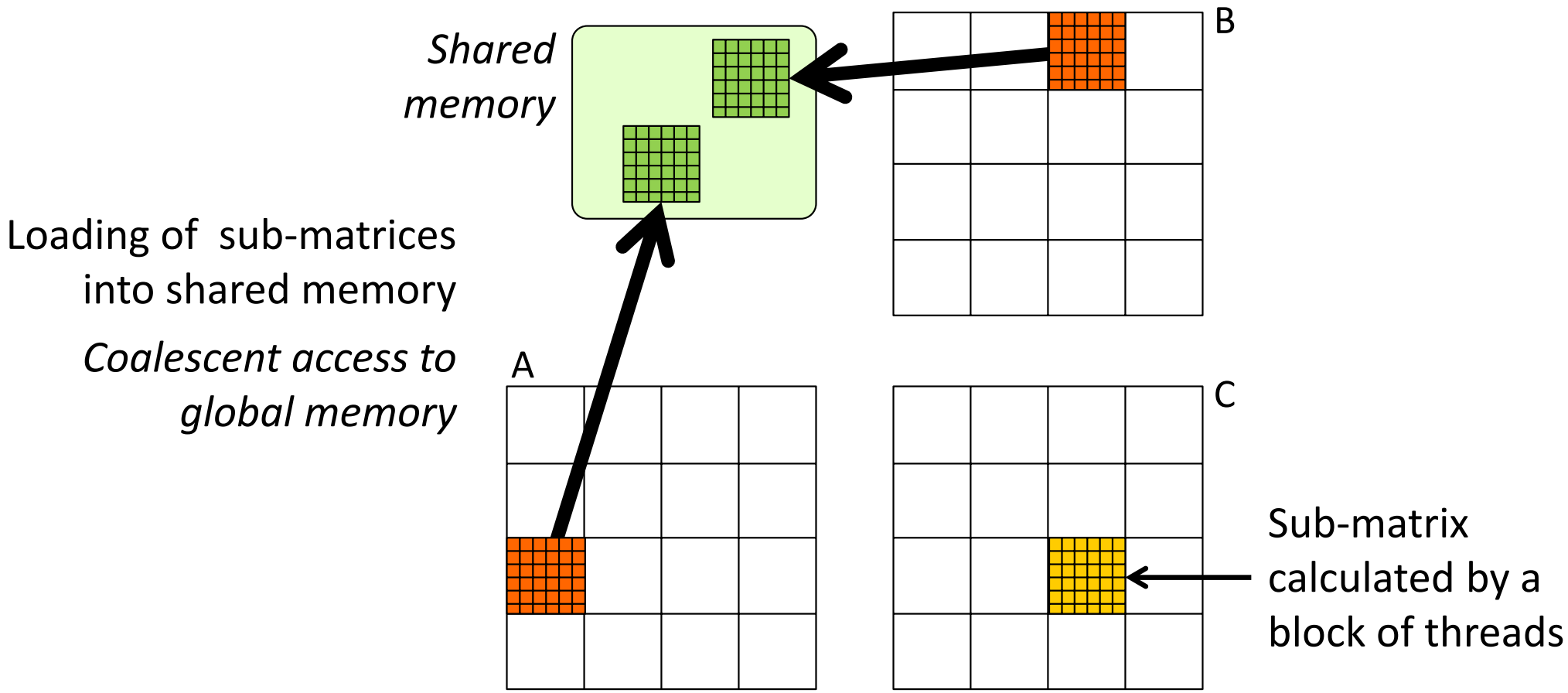
CentraleSupélec université PARIS-SACLAY
ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)
LISN LABORATOIRE INTERDISCIPLINAIRE DES SCIENCES DU NUMÉRIQUE

Ex1: 2D grid of 2D square blocks

Product of matrices of *n×n* elements

Step 0.a

- 2D-blocks of $B_{xy} \times B_{xy}$ threads (kernel k2)
- With: $n = k.B_{xy}$ $(k \in \mathbb{N})$

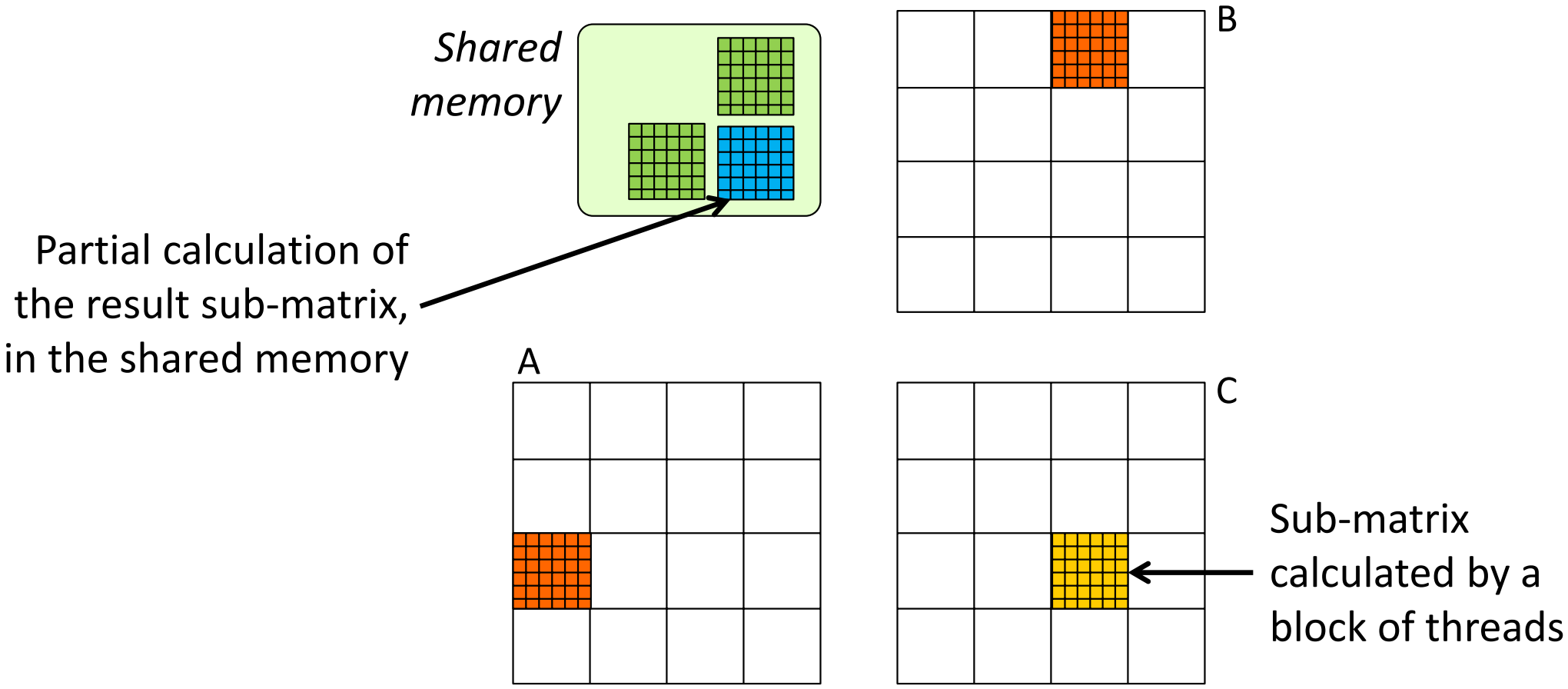


Ex1: 2D grid of 2D square blocks

Product of matrices of *n×n* elements

Step 0.b

- 2D-blocks of $B_{xy} \times B_{xy}$ threads (kernel k2)
- With: $n = k.B_{xy}$ $(k \in \mathbb{N})$

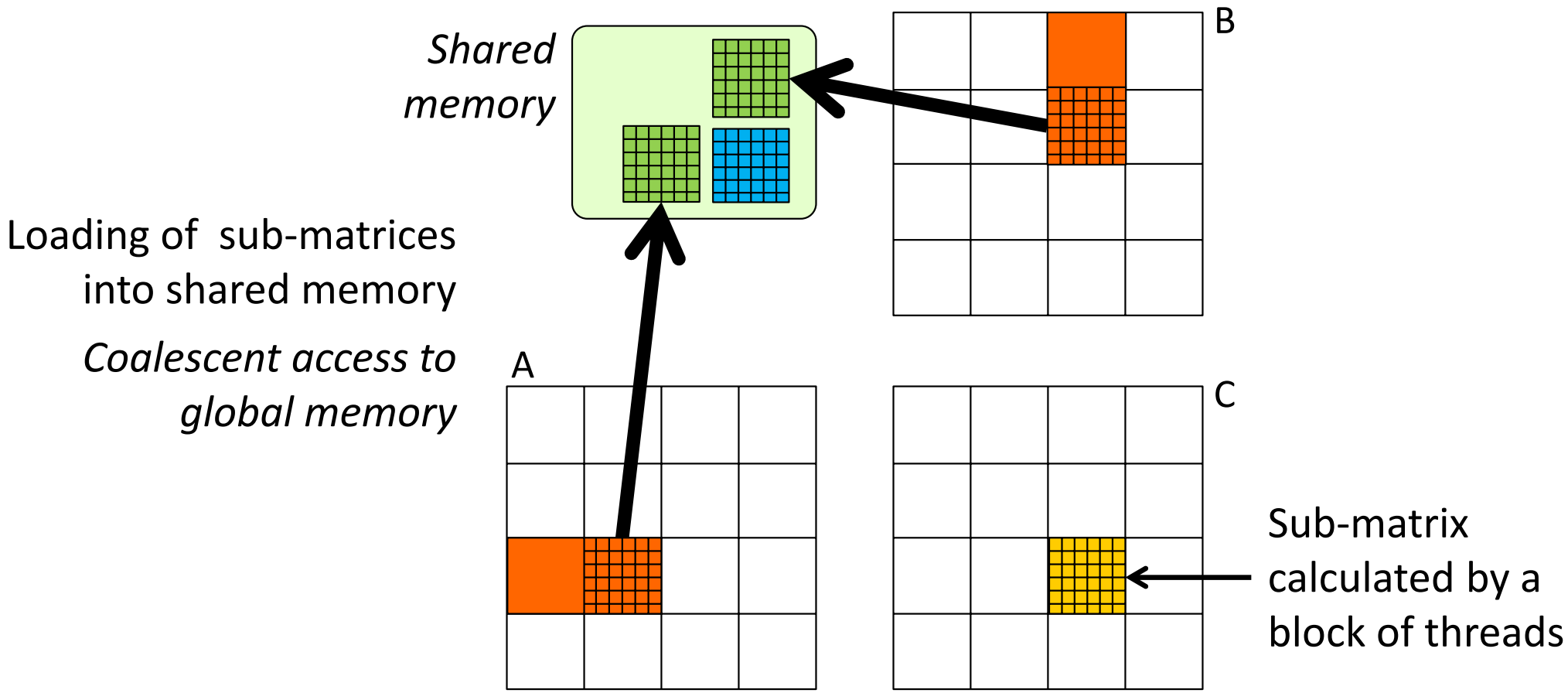


Ex1: 2D grid of 2D square blocks

Product of matrices of *n×n* elements

Step 1.a

- 2D-blocks of $B_{xy} \times B_{xy}$ threads (kernel k2)
- With: $n = k.B_{xy}$ $(k \in \mathbb{N})$

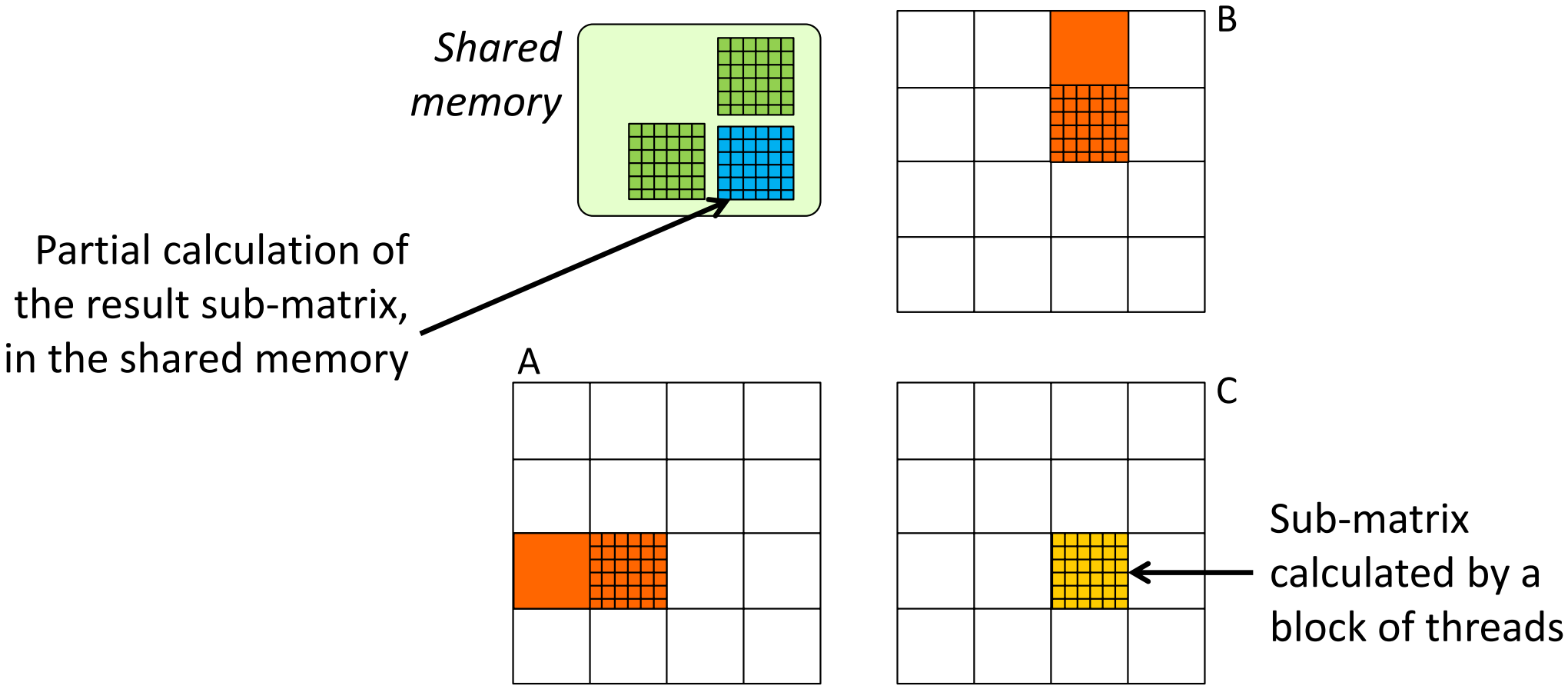


Ex1: 2D grid of 2D square blocks

Product of matrices of *n×n* elements

Step 1.b

- 2D-blocks of $B_{xy} \times B_{xy}$ threads (kernel k2)
- With: $n = k.B_{xy}$ $(k \in \mathbb{N})$

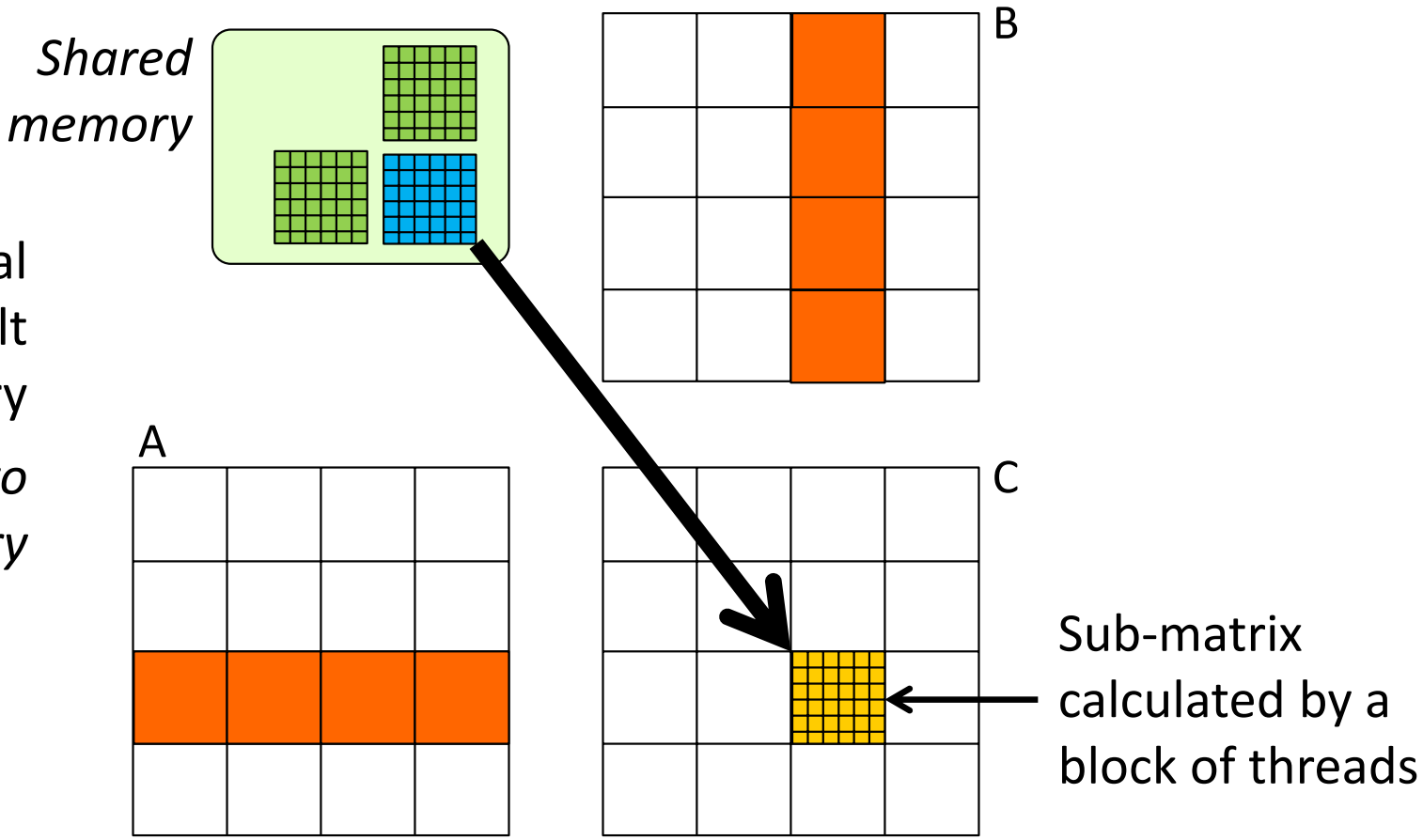


Ex1: 2D grid of 2D square blocks

Product of matrices of *n×n* elements

Last step

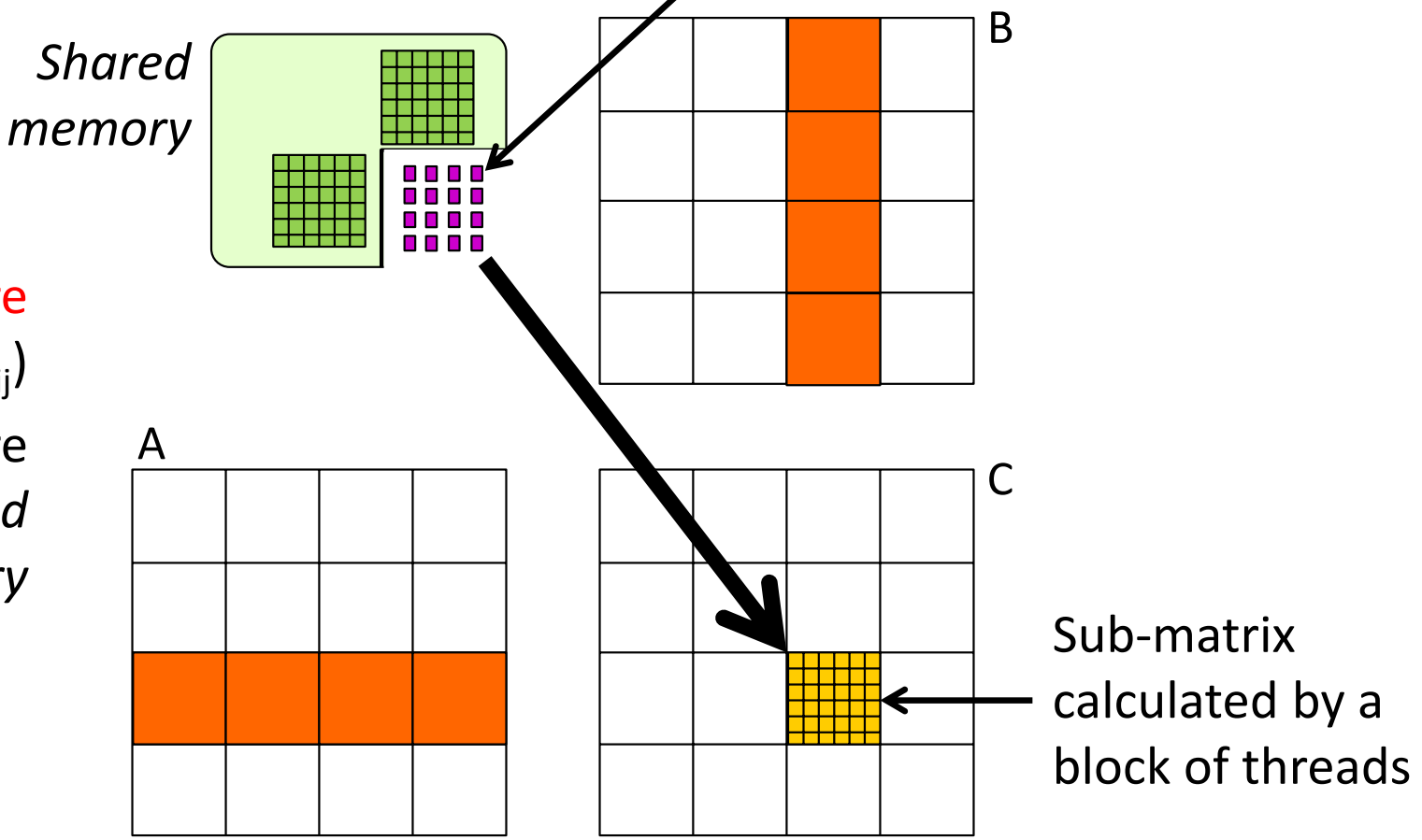
- 2D-blocks of $B_{xy} \times B_{xy}$ threads (kernel k2)
- With: $n = k.B_{xy}$ $(k \in \mathbb{N})$



Ex1: 2D grid of 2D square blocks

Product of matrices of *n×n* elements

- 2D-blocks of $B_{xy} \times B_{xy}$ threads (kernel k2)
- With: $n = k.B_{xy}$ $(k \in \mathbb{N})$



Threads do not share their results (C_{ij})

→ No need to store them in *shared memory*

Ex1: 2D grid of 2D square blocks

Product of matrices of *n×n* elements

- 2D-blocks of $B_{xy} \times B_{xy}$ threads (kernel k2)
- With: $n = k.B_{xy}$ $(k \in \mathbb{N})$

Q1.1 - Define the 2D-grid of 2D-blocks

Q1.2 - Design the algorithm and the code of the kernel k2

- Declaration of variables in Shared Memory
- Computation loop (with synchronizations)
  - Caching of data
  - Calculations
- Storing results in global memory

Q1.3 - Check the coalescence of memory accesses

Ex1: 2D grid of 2D square blocks

Product of matrices of *n×n* elements

- 2D-blocks of $B_{xy} \times B_{xy}$ threads (kernel k2)
- With: $n = k.B_{xy}$ $(k \in \mathbb{N})$

Q1.4 - Compute the total number of memory accesses <u>requested</u> by the *threads:*

$$N_{RAM\ accesses}^{requested\ by\ all\ threads} = (n_{threads}.n_{RAM\ accesses}^{requested\ by\ 1\ thread})$$

Compute the gain compared to the version without shared memory

Q1.5 - Compute the total number of memory accesses <u>achieved</u> by the *warps.*

With coalescent accesses: 1 *warp* accesses 32 data in $t_{1\ RAM\ access}$

Model : $T_{RAM\ access}^{total} = N_{RAM\ accesses}^{achieved\ by\ all\ warps}.t_{1\ RAM\ access}$

Compute the gain compared to the version without shared memory

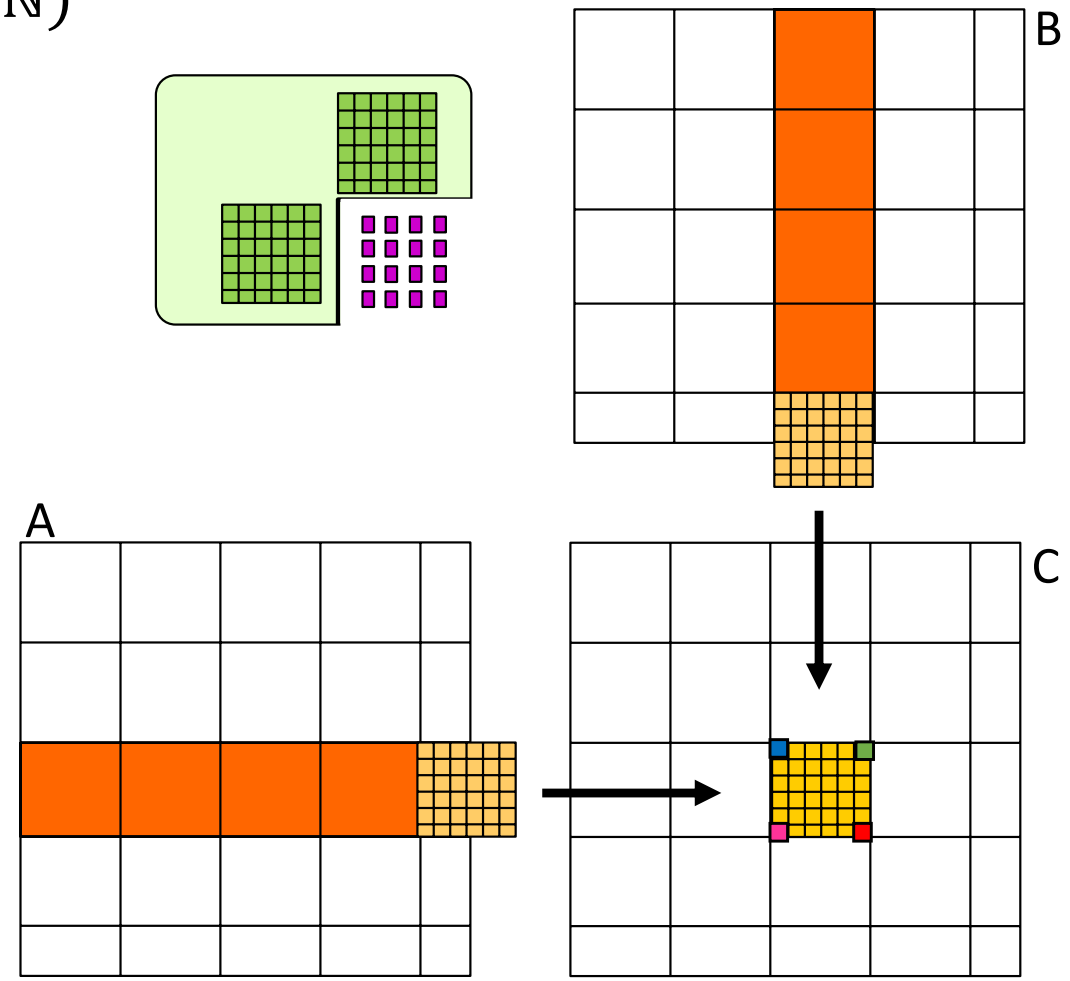
Ex2: 2D grid of 2D square blocks - Boundaries

Product of matrices of *n×n* elements

- 2D-blocks of $B_{xy} \times B_{xy}$ threads (kernel k3)
- With: $n \neq k.B_{xy}$ $(k \in \mathbb{N})$

What are the conditions for:

- loading A into shared memory?
- loading B into shared memory?
- doing the calculations and writing to C?



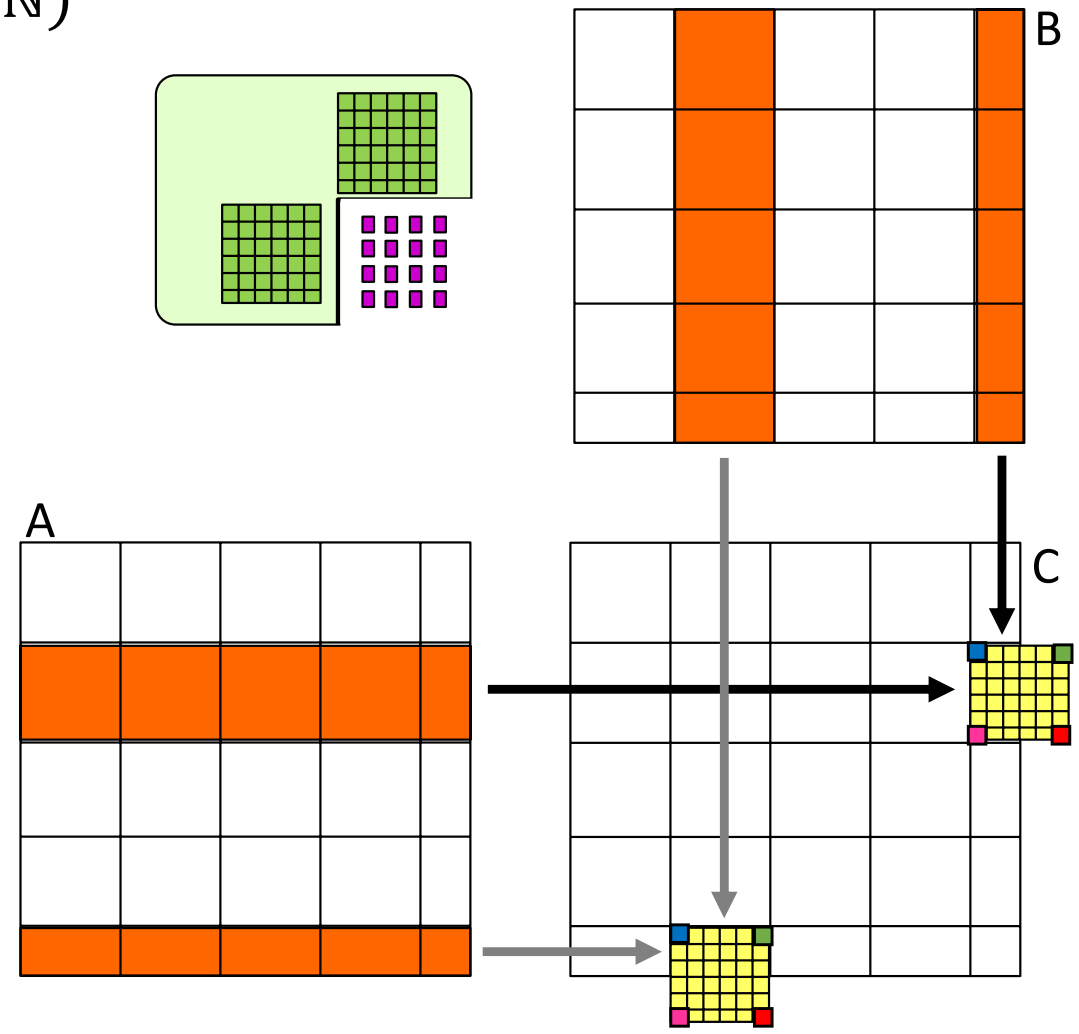
Ex2: 2D grid of 2D square blocks - Boundaries

Product of matrices of *n*×*n* elements

- 2D-blocks of $B_{xy} \times B_{xy}$ threads (kernel k3)
- With: $n \neq k.B_{xy}$ $(k \in \mathbb{N})$

What are the conditions for:

- loading A into shared memory?
- loading B into shared memory?
- doing the calculations and writing to C?



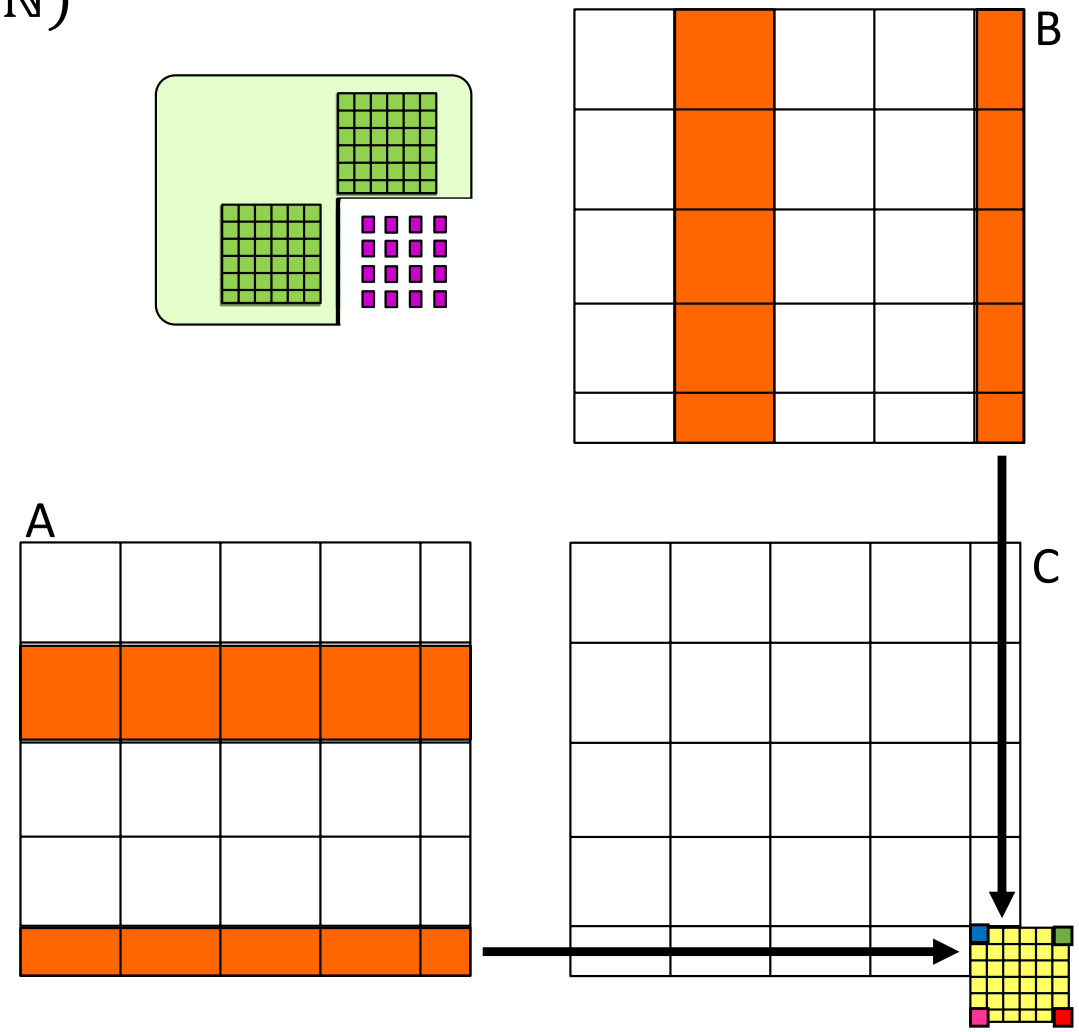
Ex2: 2D grid of 2D square blocks - Boundaries

Product of matrices of *n×n* elements

- 2D-blocks of $B_{xy} \times B_{xy}$ threads (kernel k3)
- With: $n \neq k.B_{xy}$ $(k \in \mathbb{N})$

What are the conditions for:

- loading A into shared memory?
- loading B into shared memory?
- doing the calculations and writing to C?



Ex2: 2D grid of 2D square blocks - Boundaries

Product of matrices of *n×n* elements

- 2D-blocks of $B_{xy} \times B_{xy}$ threads (kernel k3)
- With: $n \neq k.B_{xy}$ $(k \in \mathbb{N})$

Q2.1 - Define the 2D-grid of 2D-blocks

Q2.2 - Design a first algorithm and code for the kernel k3:

- Strategy 1: insert 0.0's into the caches of A and B when there is no data to store there
- Implement all necessary tests, and only the necessary tests! (unnecessary tests could slow down the application)

Q2.3 - Design a second algorithm and code for the kernel k3:

- Strategy 2: Do not insert dummy data into the caches of A and B, but adapt the tests and calculations that follow

POLYTECH
PARIS-SACLAY

TD3: Dense matrix product on GPU using shared memory

End