Mitglied der Helmholtz-Gemeinschaft

JÜLICH
FORSCHUNGSZENTRUM

DGEMM
Tiled matrix multiplication with Cuda

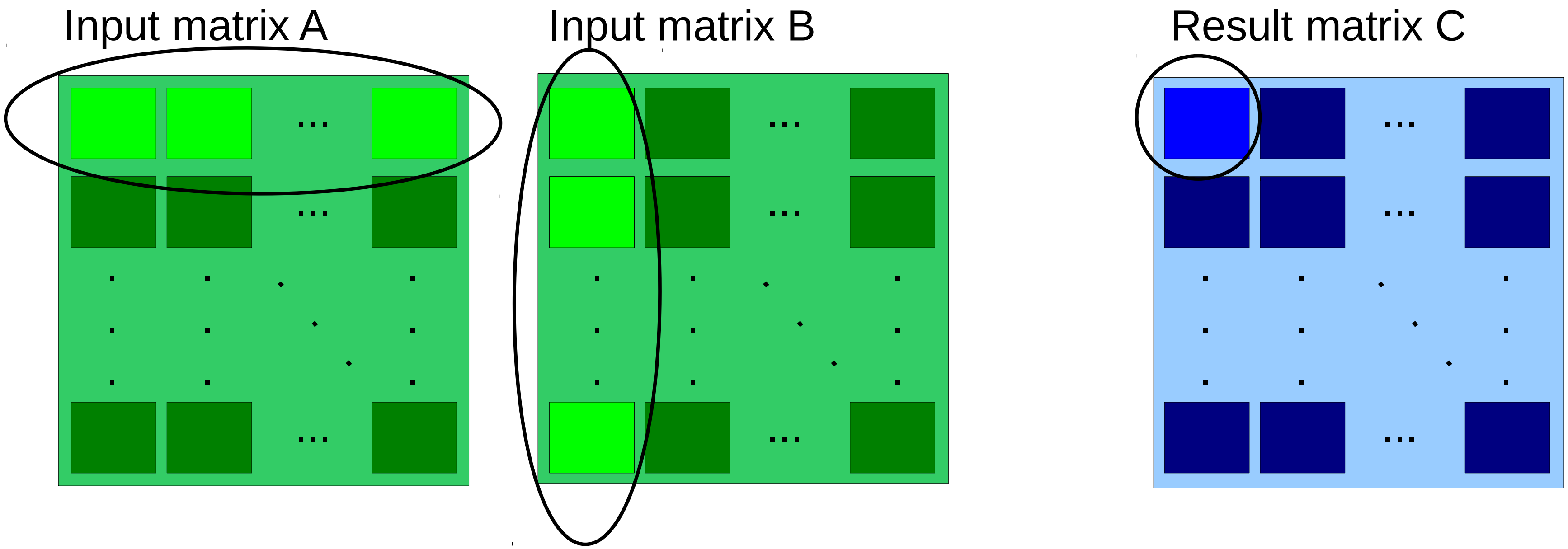
Jochen Kreutz (JSC)

GPUProgramming with CUDA @ JSC, 25. - 27. April 2016

Overview

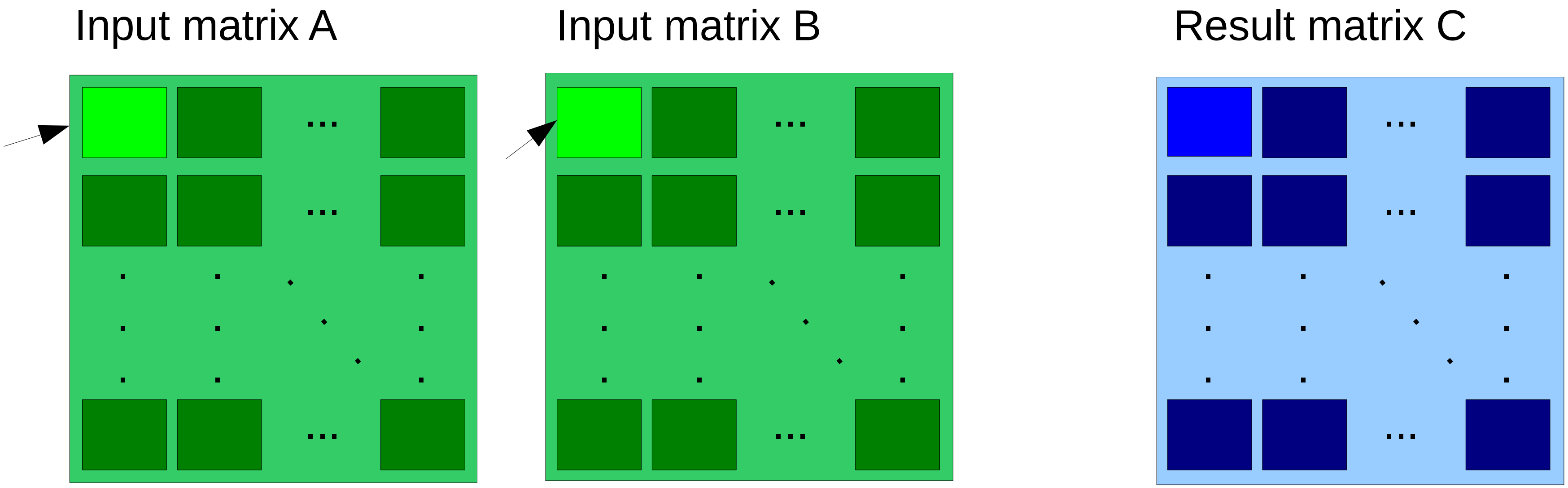
- Tiled matrix multiplication algorithm
- Cuda implementation with and without streams
- Using multi-GPUs and streams

Tiled Matrix Multiplication



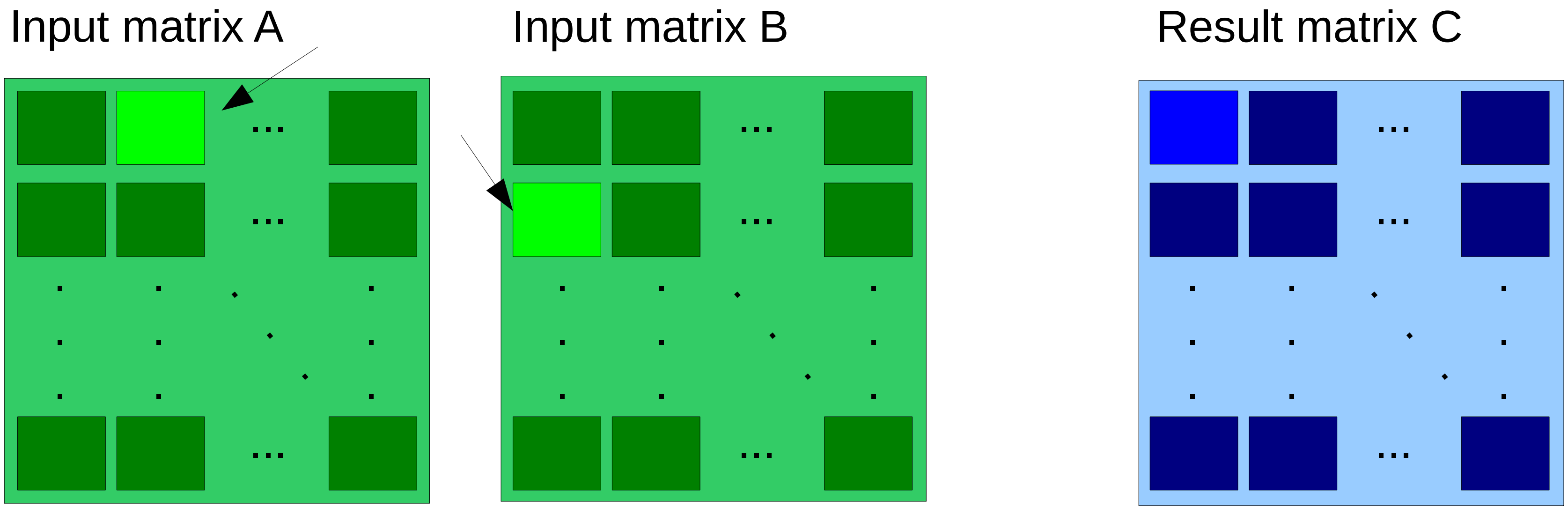
- Split matrices into tiles
- Allows for distributing work onto different streams (and GPUs)

Tiled Matrix Multiplication



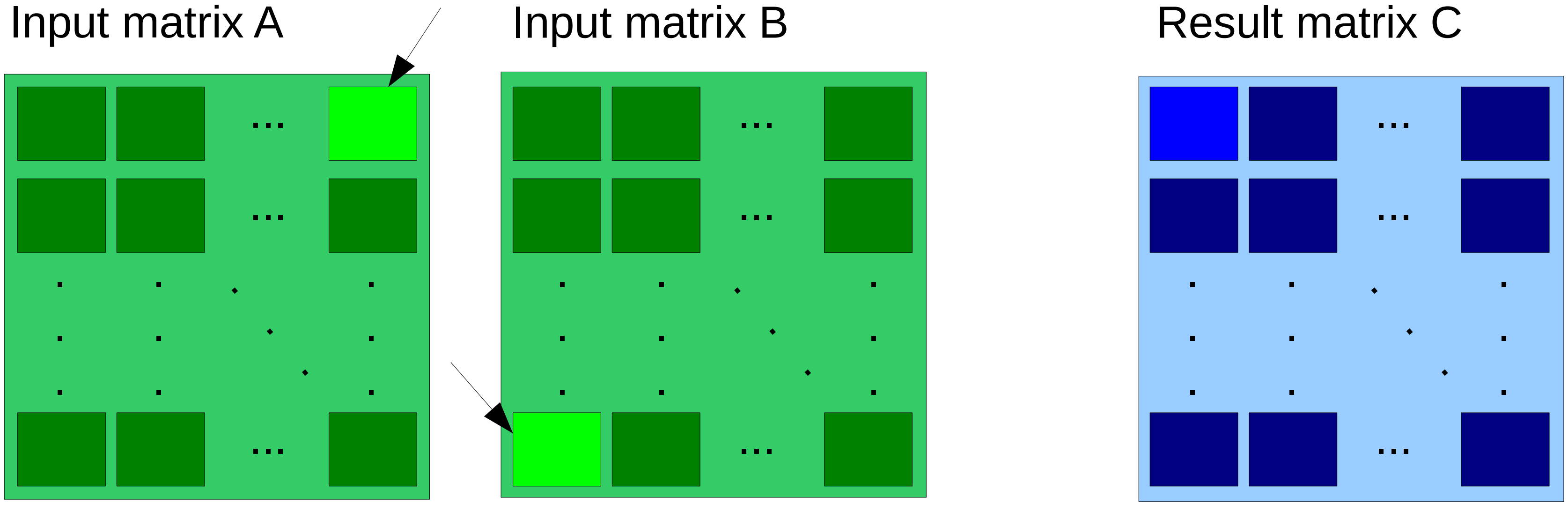
- Do partial (block-wise) computation
- Sum up partial results

Tiled Matrix Multiplication



- Do partial (block-wise) computation
- Sum up partial results

Tiled Matrix Multiplication

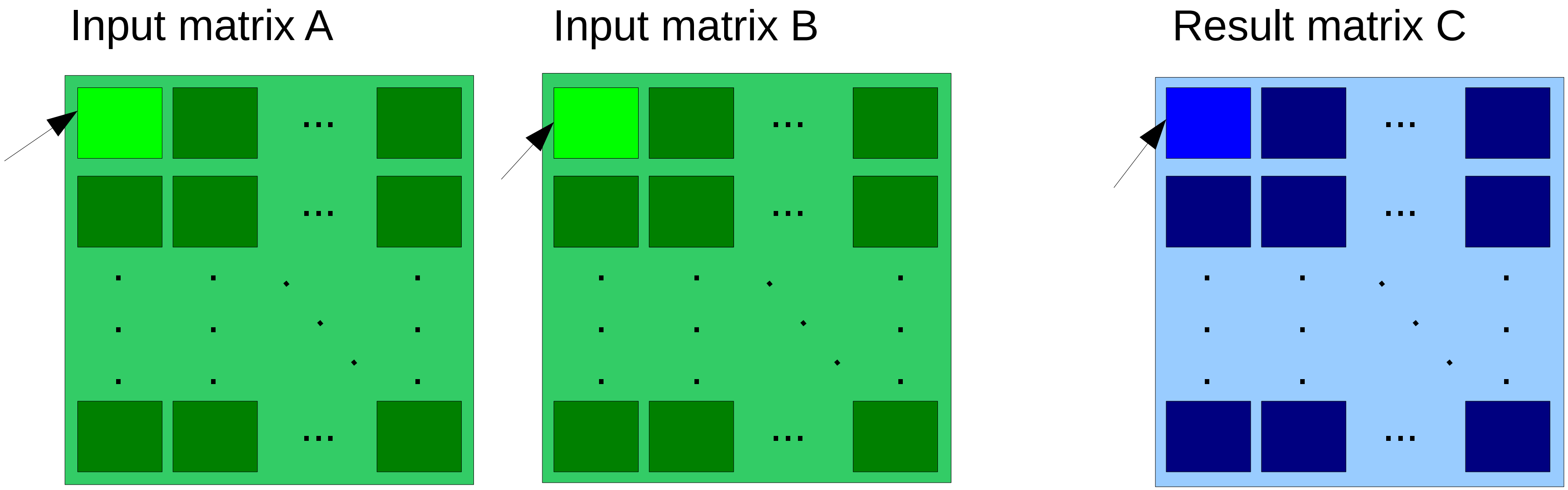


- Do partial (block-wise) computation
- Sum up partial results

Tiled Matrix Multiplication

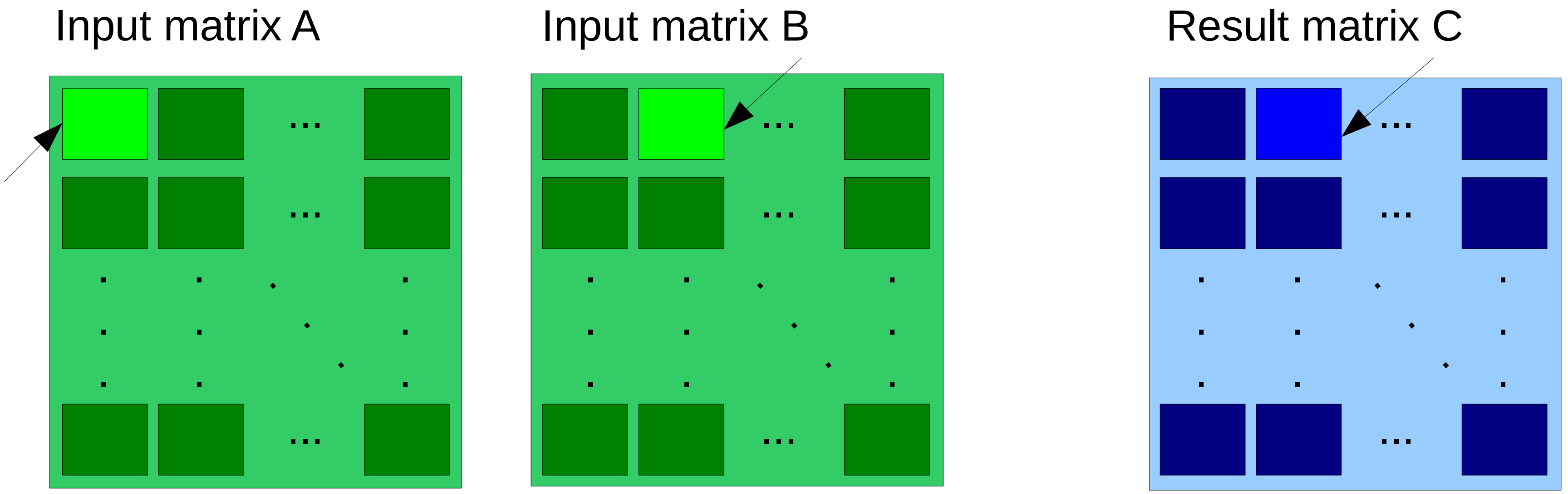
- Change order of computations and run over all tiles of the result matrix in an inner loop
- Do first computations for all tiles in result matrix and then repeat with next tiles of input matrices
- Allows for concurrency in computation of tiles in C

Tiled Matrix Multiplication



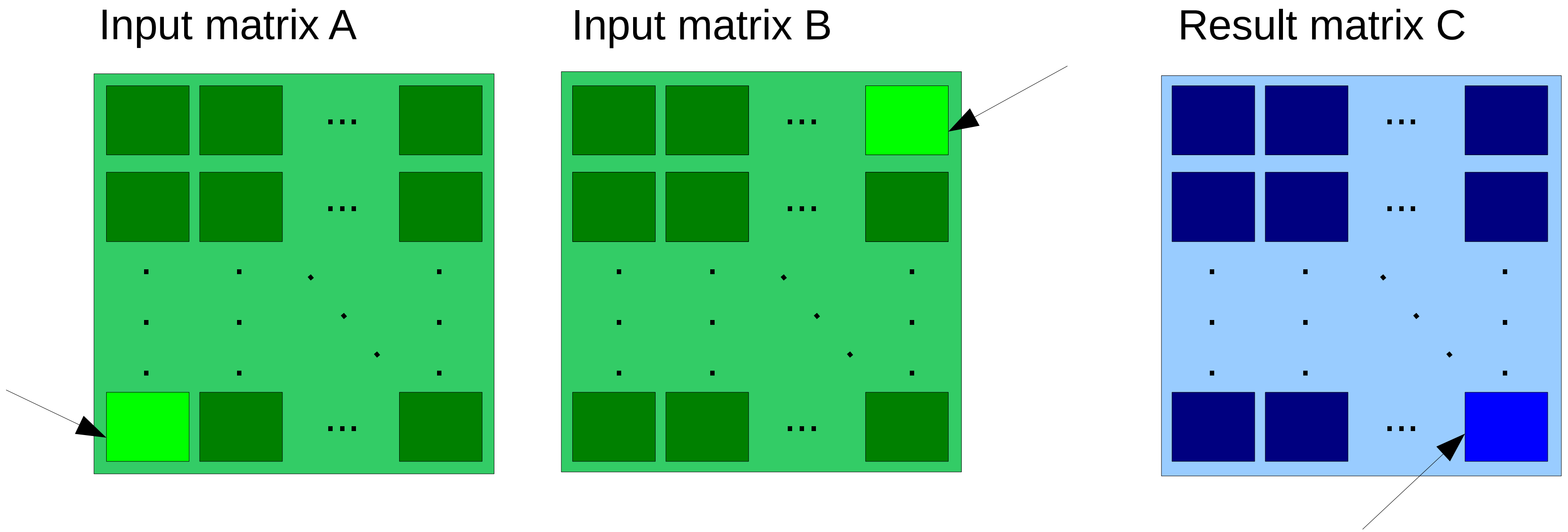
- Change order of computations and run over all tiles of the result matrix with an inner loop
- Do first computations for all tiles in result matrix and then repeat with next tiles of input matrices

Tiled Matrix Multiplication



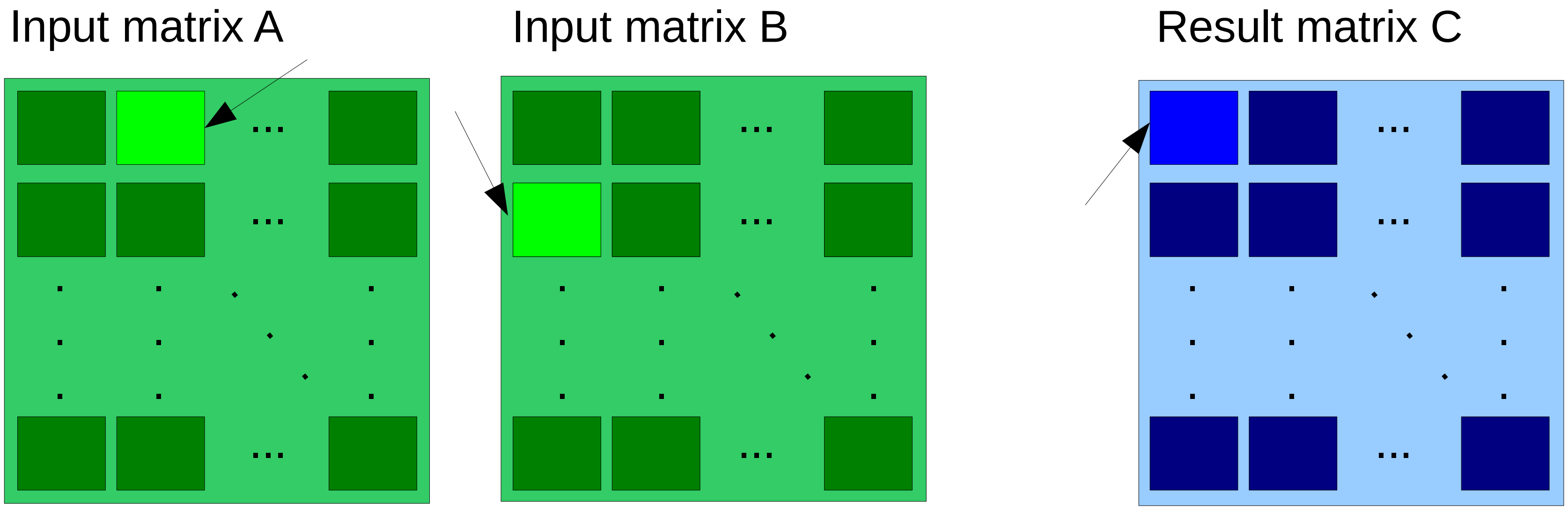
- Change order of computations and run over all tiles of the result matrix in the inner loop
- Do first computations for tiles in result matrix and then proceed to next tiles of input matrices

Tiled Matrix Multiplication



- Change order of computations and run over all tiles of the result matrix in the inner loop
- Do first computations for tiles in result matrix and then proceed to next tiles of input matrices

Tiled Matrix Multiplication



- Change order of computations and run over all tiles of the result matrix in the inner loop
- Do first computations for tiles in result matrix and then proceed to next tiles of input matrices

Tiled Matrix Multiplication - Implementation

Loop over tiles

```
// loop over inner tile dimension
    for ( int iktile = 0; iktile < ntiles; iktile++ ) {

      // loop over row tiles
      for ( int irowtile = 0; irowtile < ntiles; irowtile++ ) {

        // loop over column tiles
        for ( int icoltile = 0; icoltile < ntiles; icoltile++ ) {

        …

        }
      }
    }
```

Tiled Matrix Multiplication - Implementation

- Tiled approach allows to operate large matrices that would not fit into GPU memory as a whole
- For each step only 3 tiles have to be present on the device
- Use pinned memory for tiles to do asynchronous host to device copies and speed up data transfers
- Set beta to 1 in cublasDgemm call to reuse previous calculated results

DGEMM

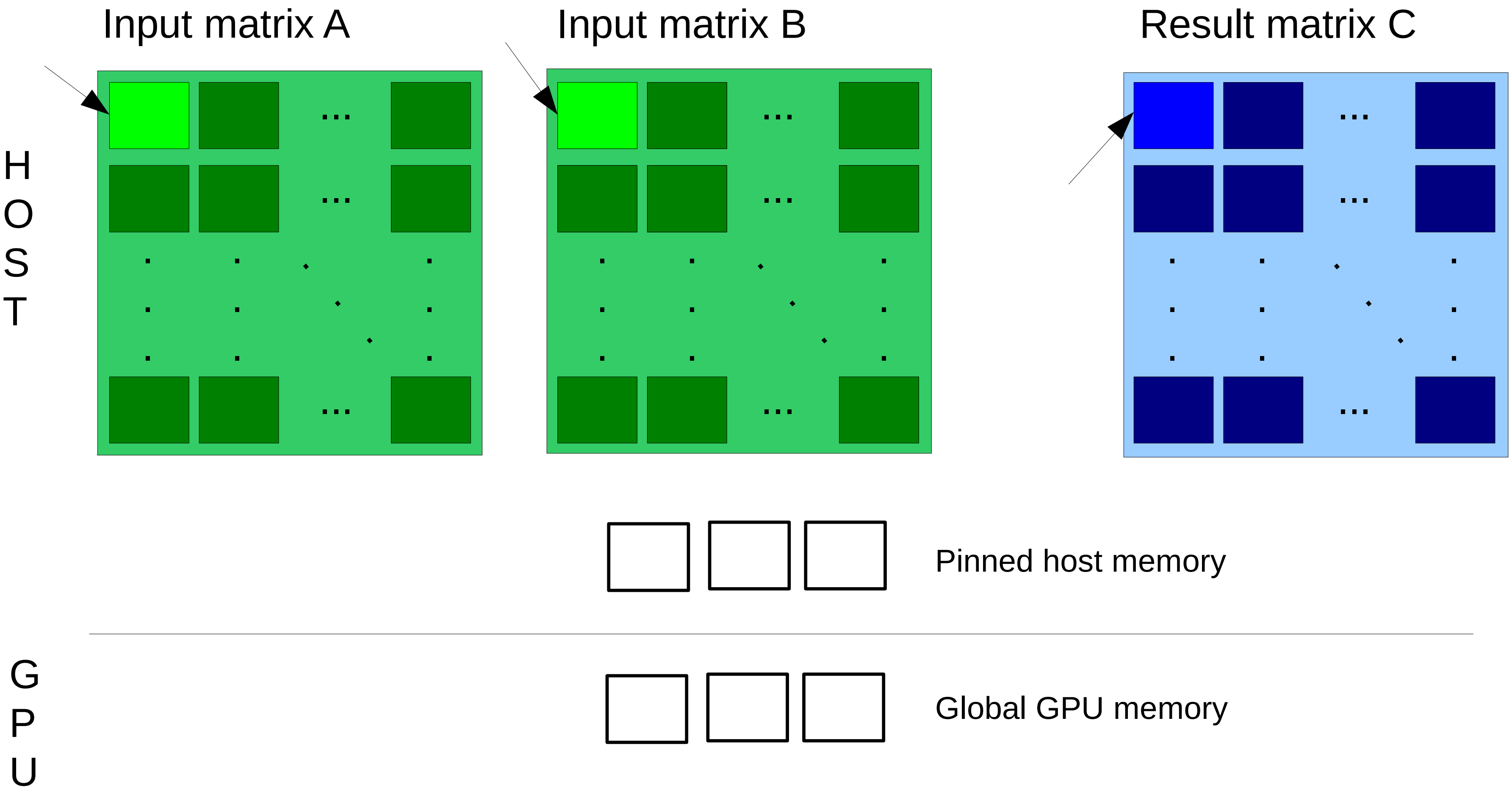
```
C := alpha*op(A)*op(B) + beta*C
```

Tiled Matrix Multiplication - Implementation

- Workflow:
  - Init data (elements of result matrix C have to be set to 0)
  - Loop over tiles in input matrices and over tiles in C

1. Read input data (3 tiles) from global matrices to pinned buffers
2. Transfer 3 relevant tiles to device
3. Call cublasDgemm with beta = 1
4. Read back results from device to pinned buffer
5. Write back temporary results (1 tile) from pinned host buffer to global result matrix in host memory

Tiled Matrix Multiplication - Implementation



Input matrix A

Input matrix B

Result matrix C

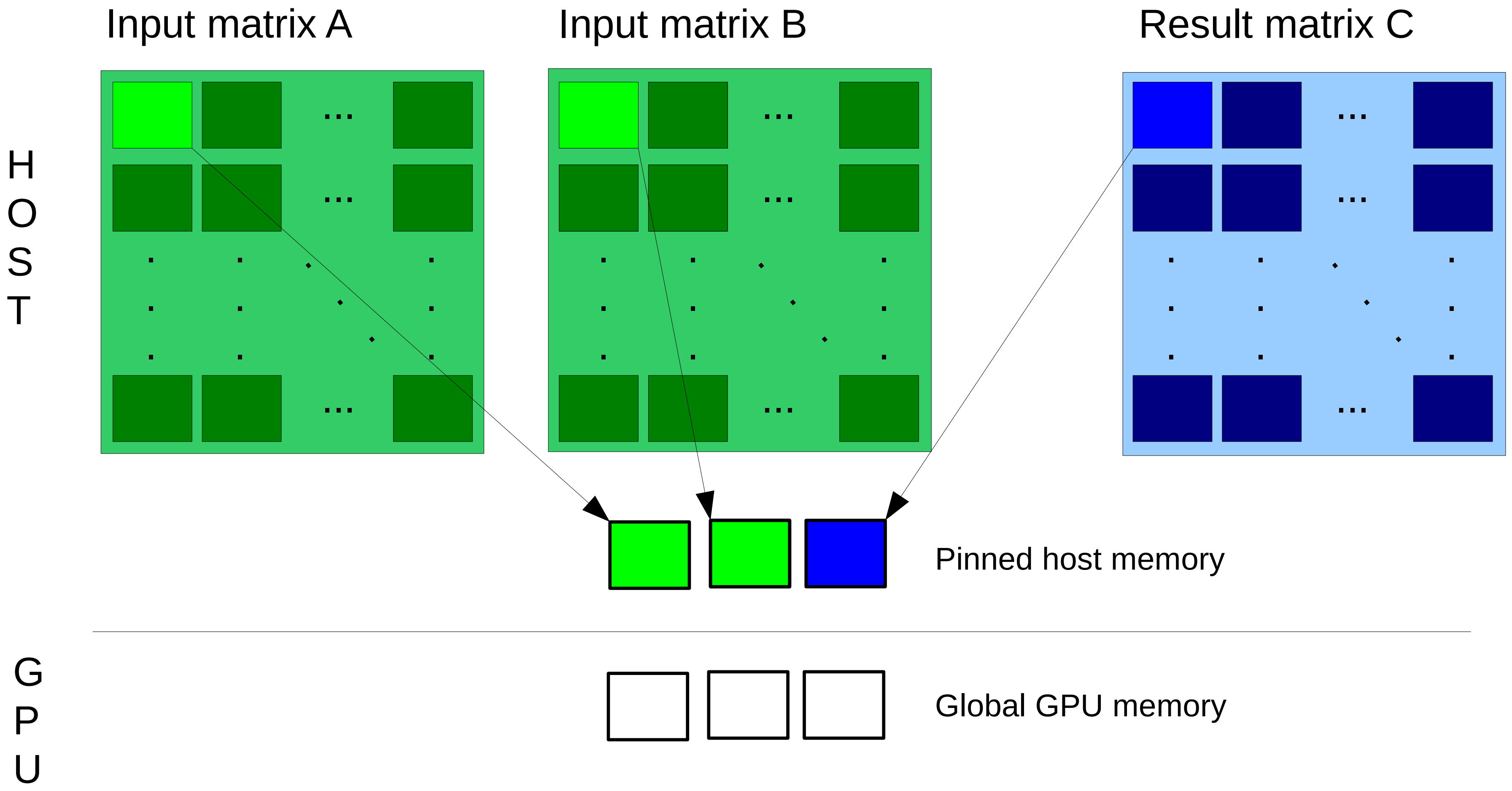
H O S T

Pinned host memory

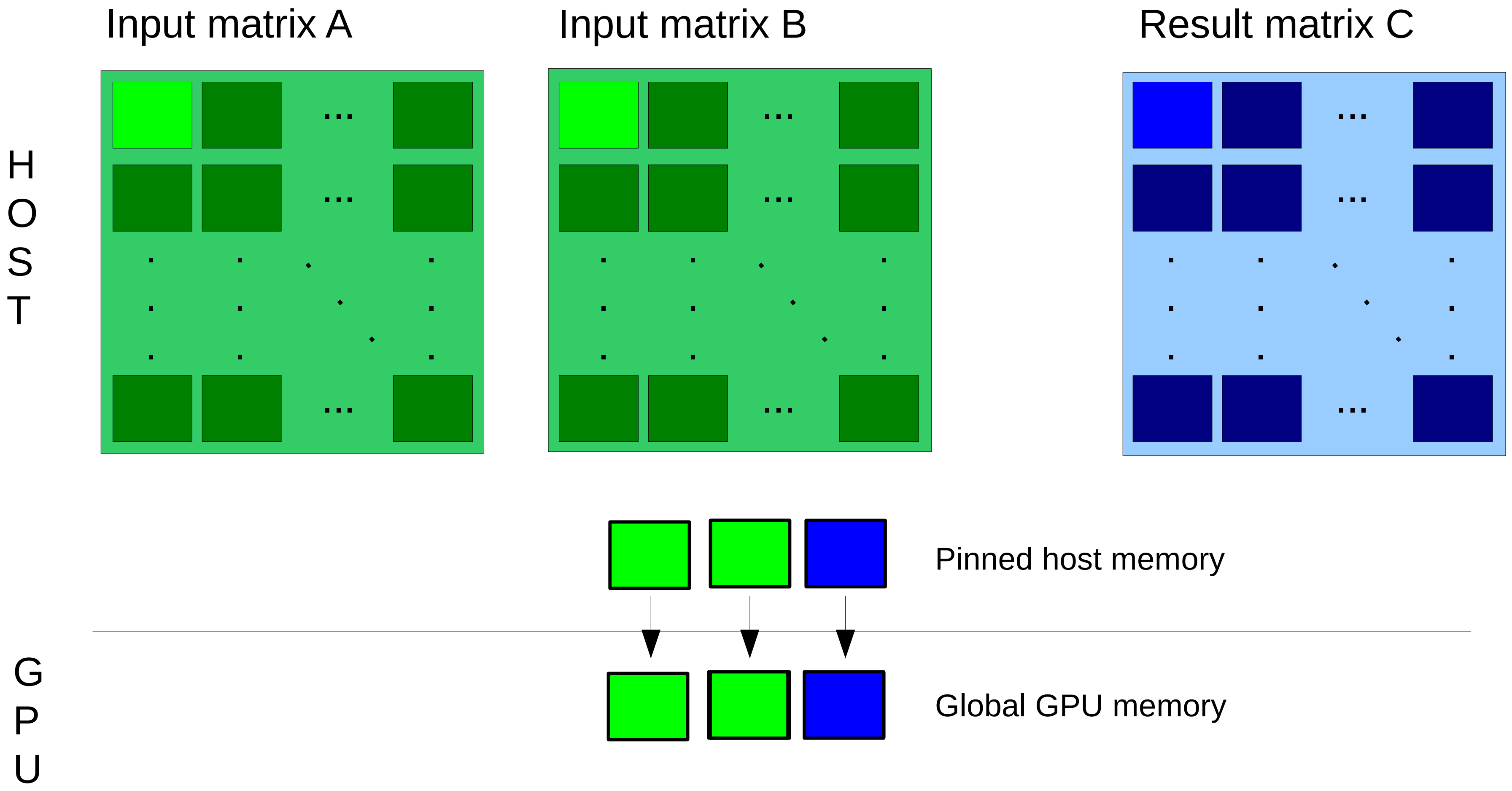
G P U

Global GPU memory

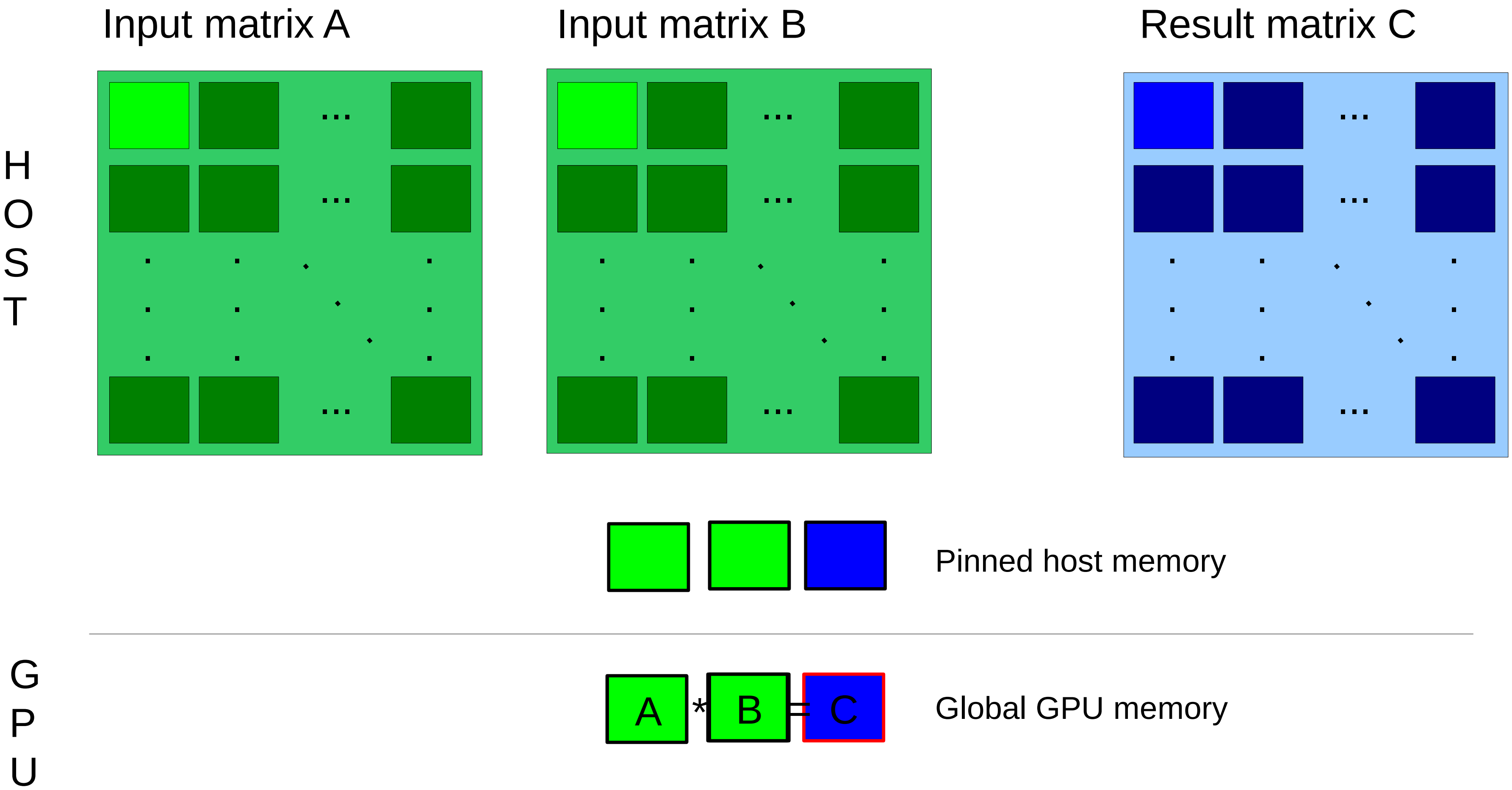
Tiled Matrix Multiplication – Implementation Step 1



Tiled Matrix Multiplication – Implementation Step 2



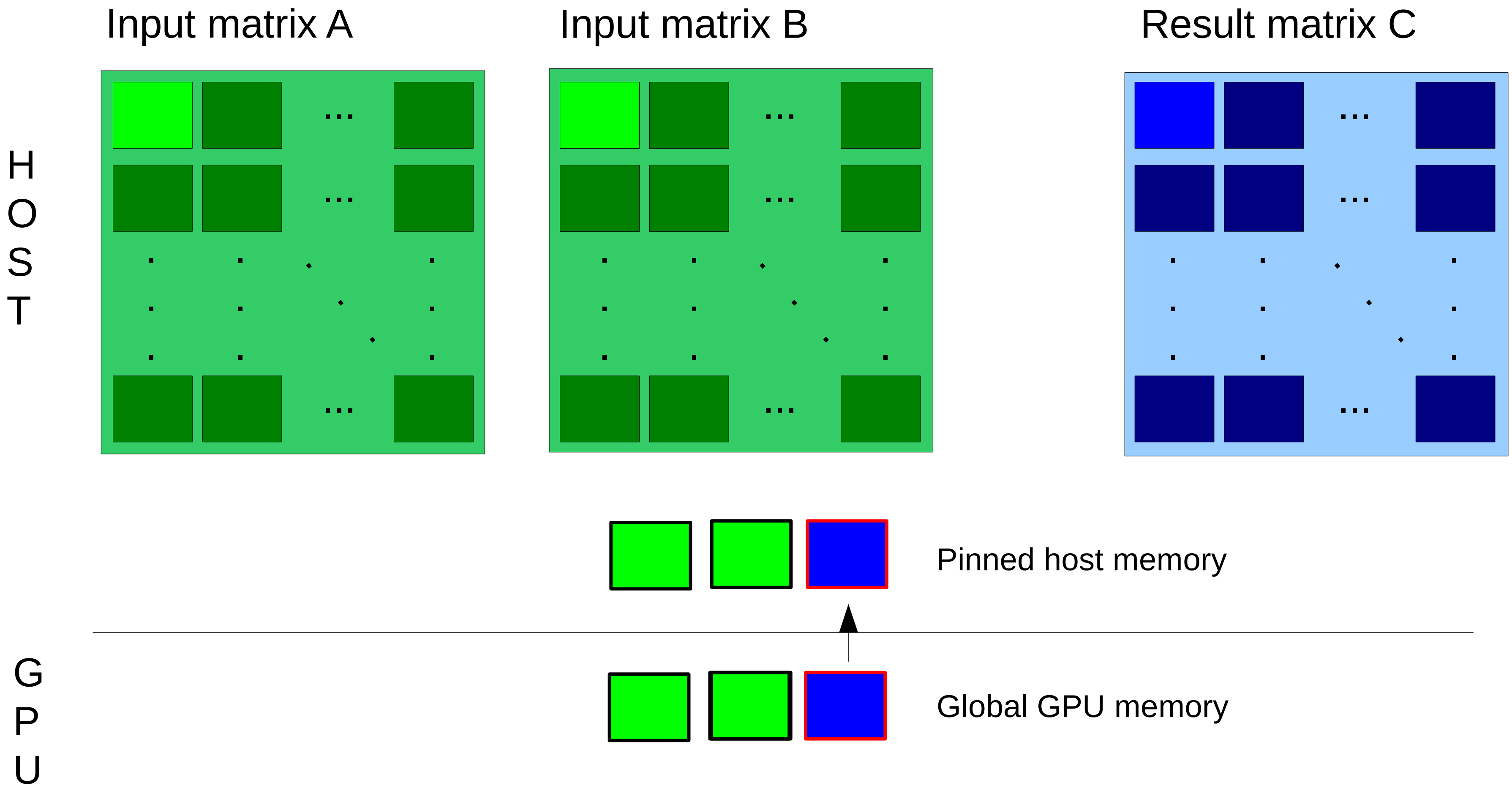
Tiled Matrix Multiplication – Implementation Step 3



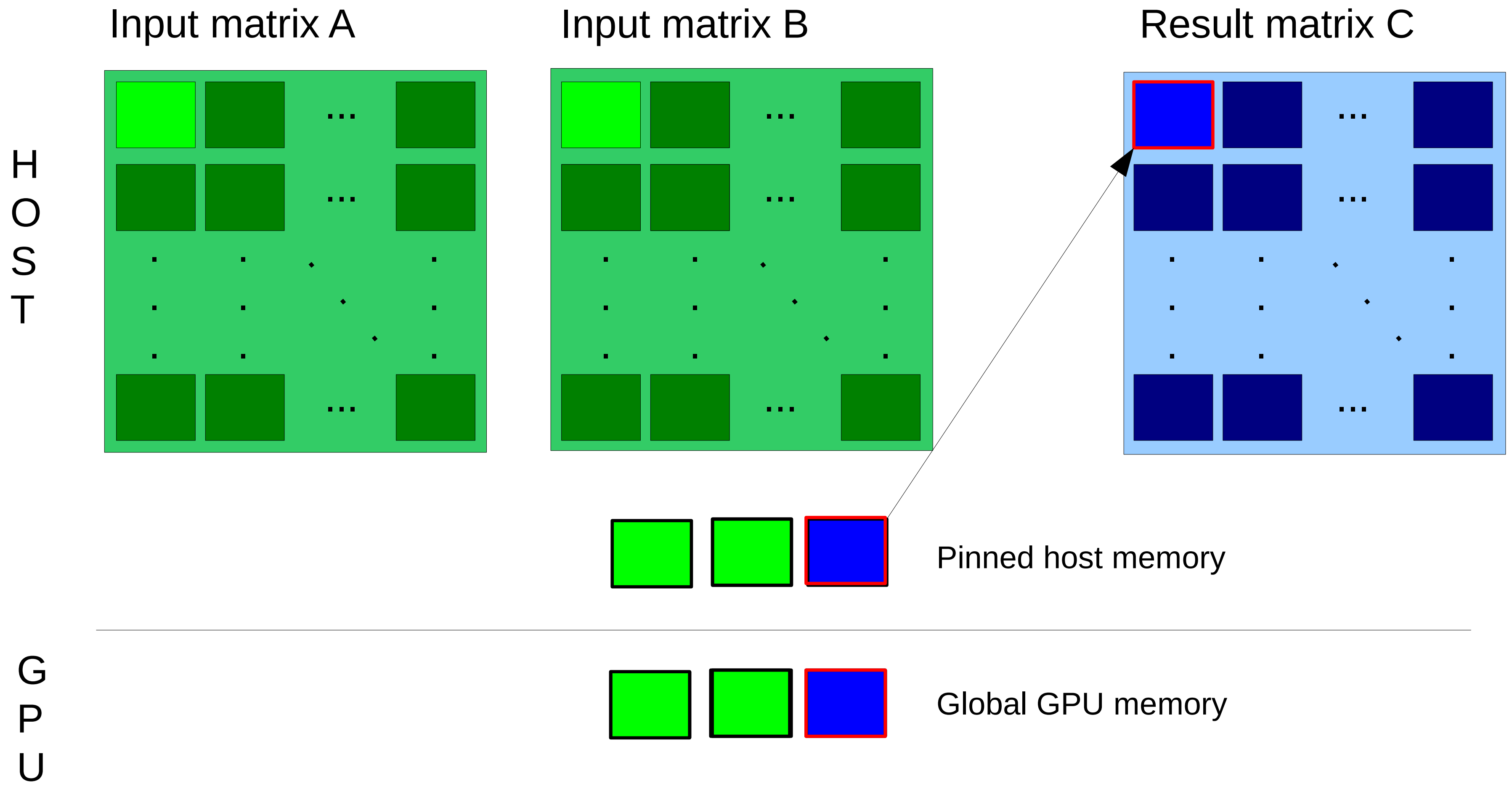
Pinned host memory

Global GPU memory

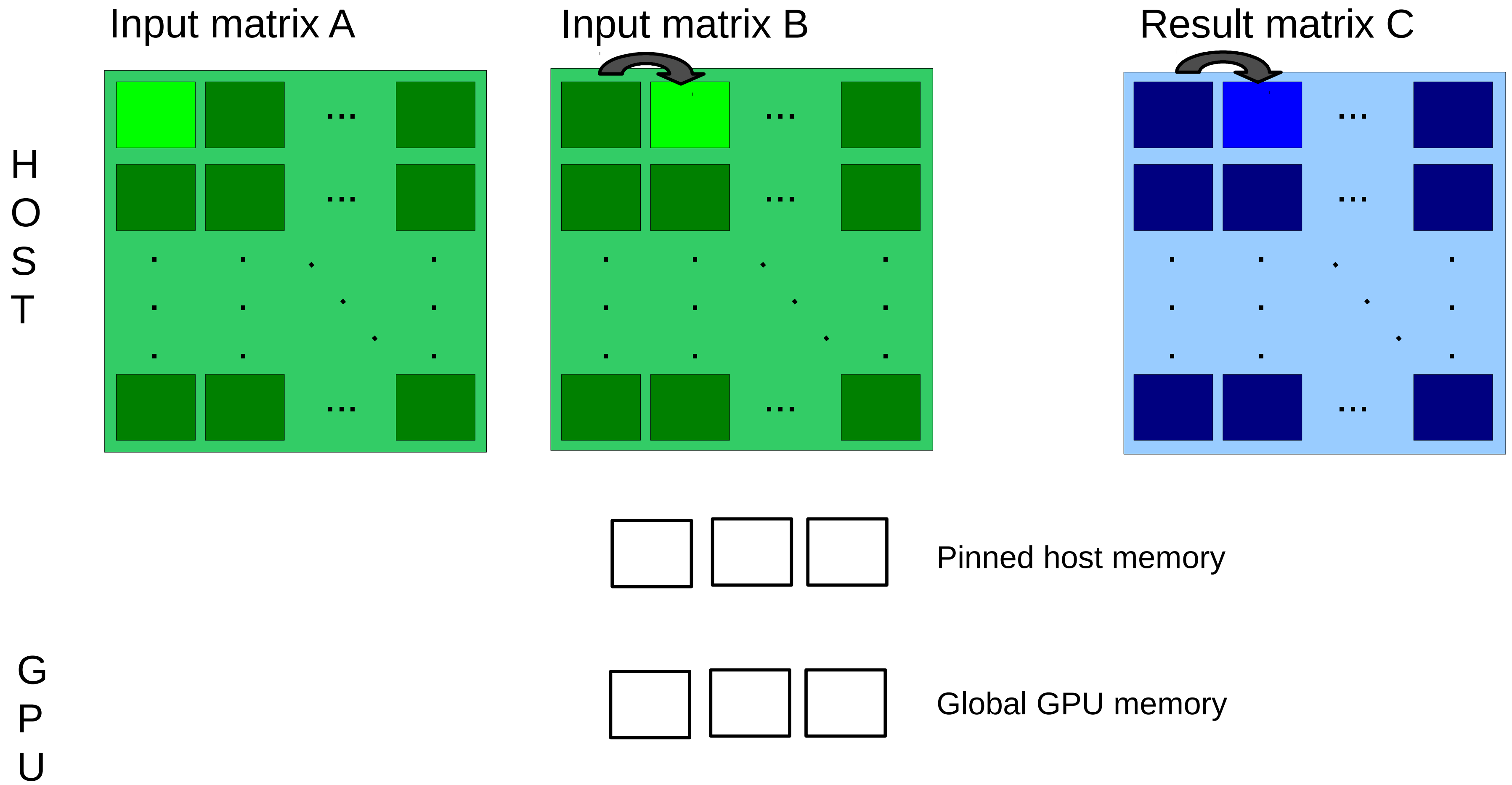
Tiled Matrix Multiplication – Implementation Step 4



Tiled Matrix Multiplication – Implementation Step 5



Tiled Matrix Multiplication – Implementation
Repeat steps 1 to 5



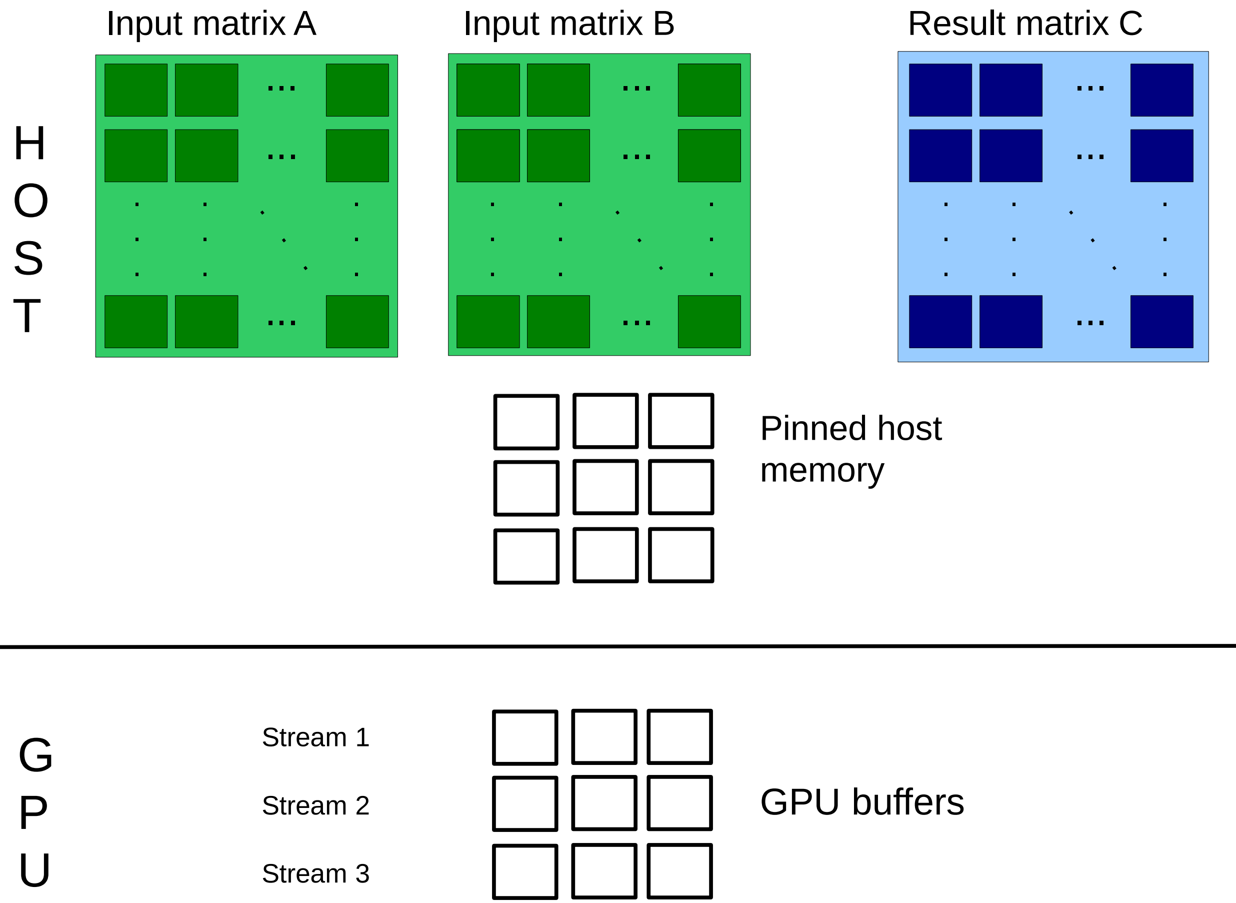
Exercise

Tiled Matrix Multiplication – Using Streams

- Distribute computation of tiles to different streams
- Use asynchronous data transfers to overlap kernel executions and memory copies
  - *Unnecessary data movement can be hidden and simplify the implementation*
- Each stream will use its own tile buffers (multi buffering)
- Synchronization will be necessary

Tiled Matrix Multiplication – Using Streams

- Example: 3 streams



Input matrix A

Input matrix B

Result matrix C

HOST

Pinned host memory

GPU

Stream 1

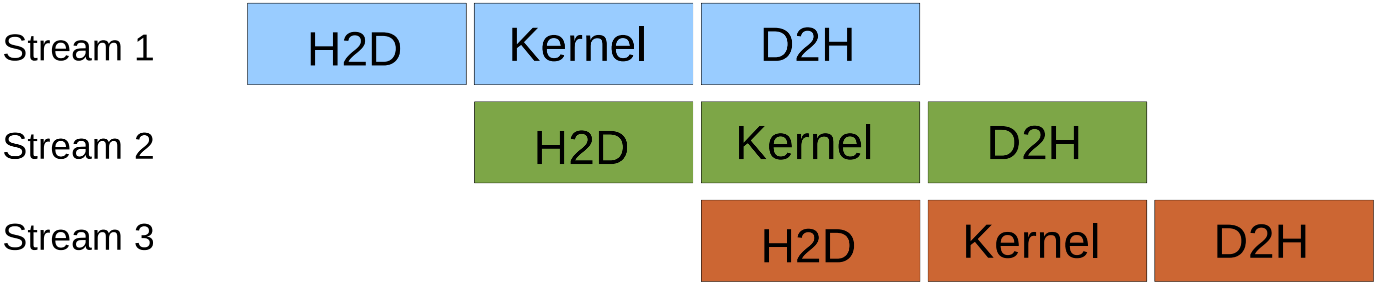
Stream 2

Stream 3

GPU buffers

Tiled Matrix Multiplication – Using Streams

- Example: 3 streams
- For every tile:
  - *H2D data transfer*
  - *Kernel execution (dgemm)*
  - *D2H data transfer*



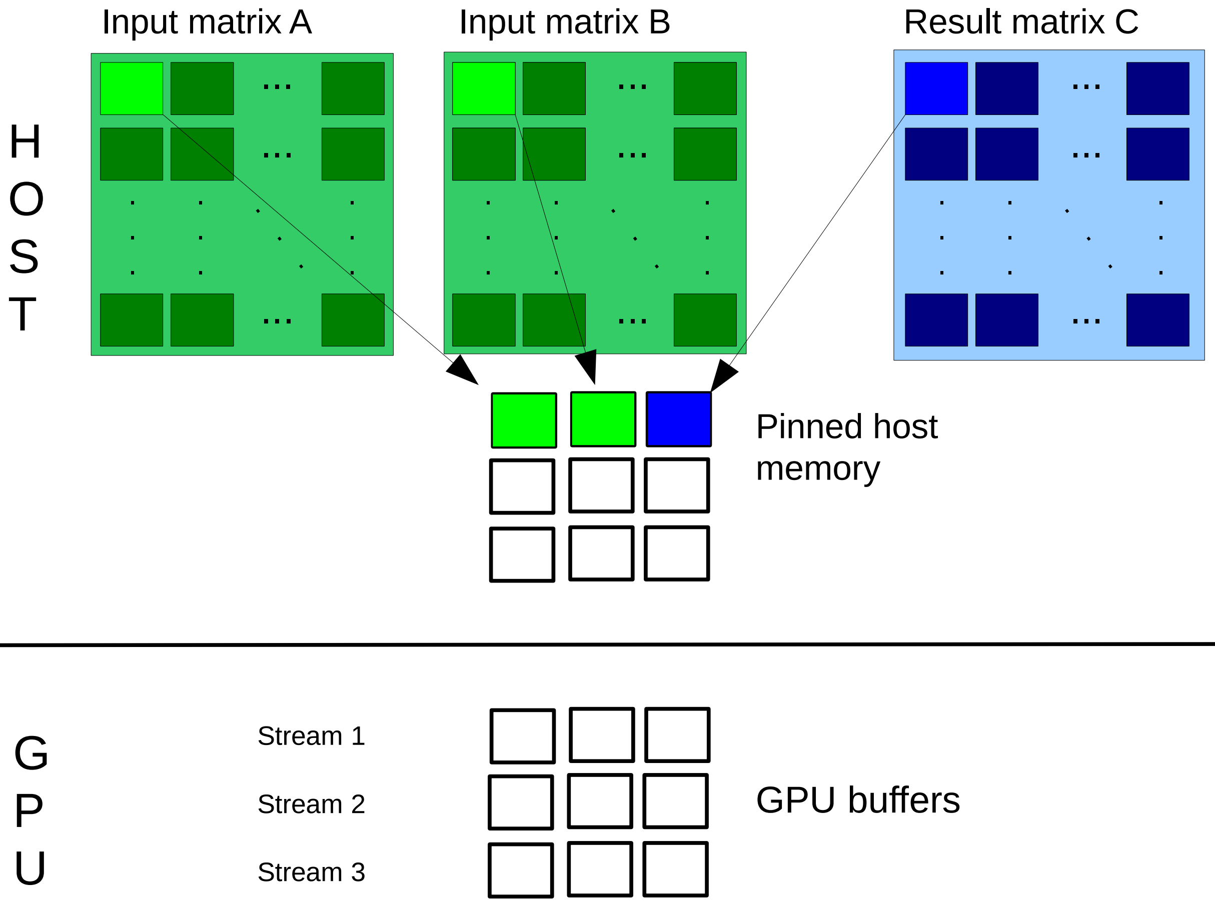
Stream 1: H2D, Kernel, D2H

Stream 2: H2D, Kernel, D2H

Stream 3: H2D, Kernel, D2H

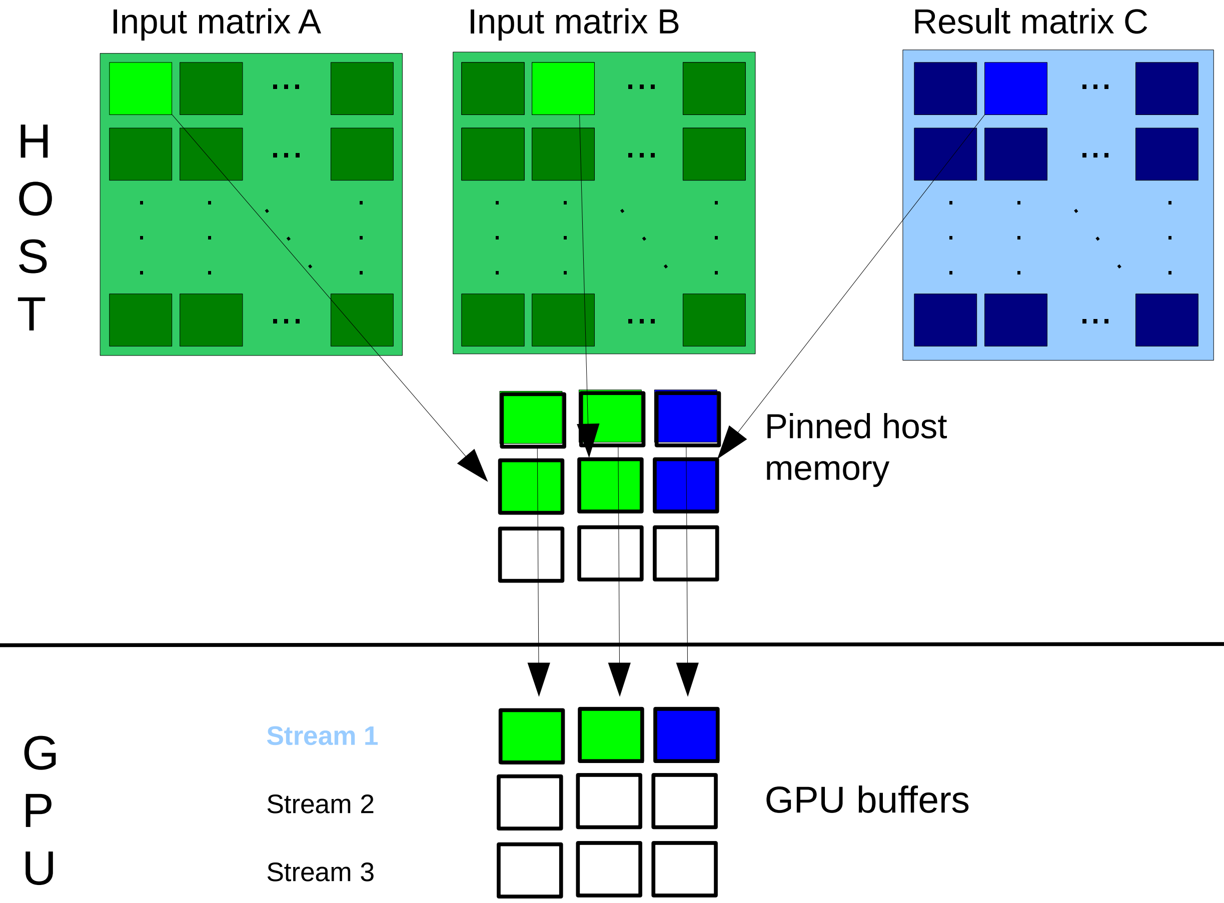
Tiled Matrix Multiplication – Using Streams

- Example: 3 streams



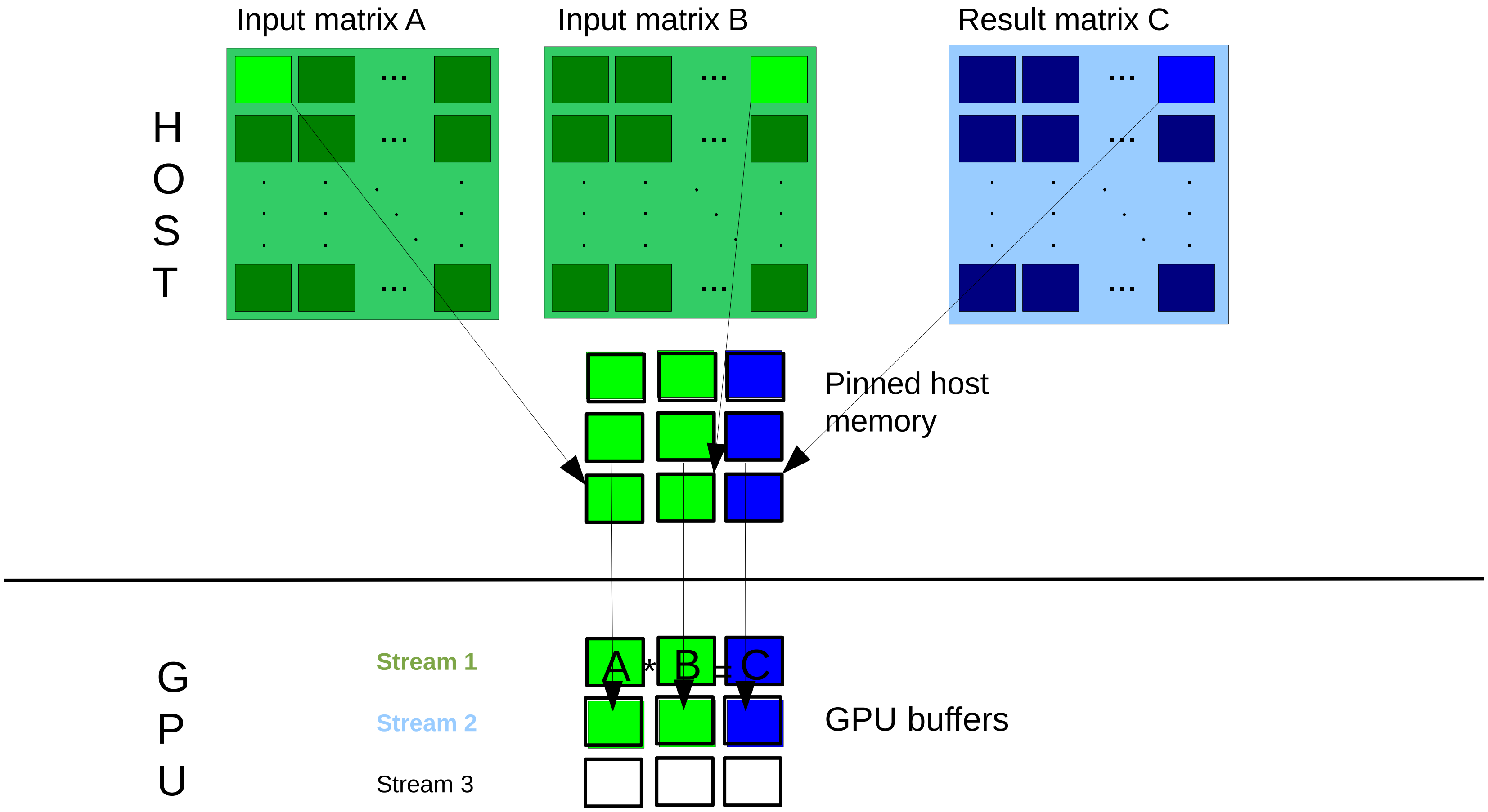
Tiled Matrix Multiplication – Using Streams

- Example: 3 streams



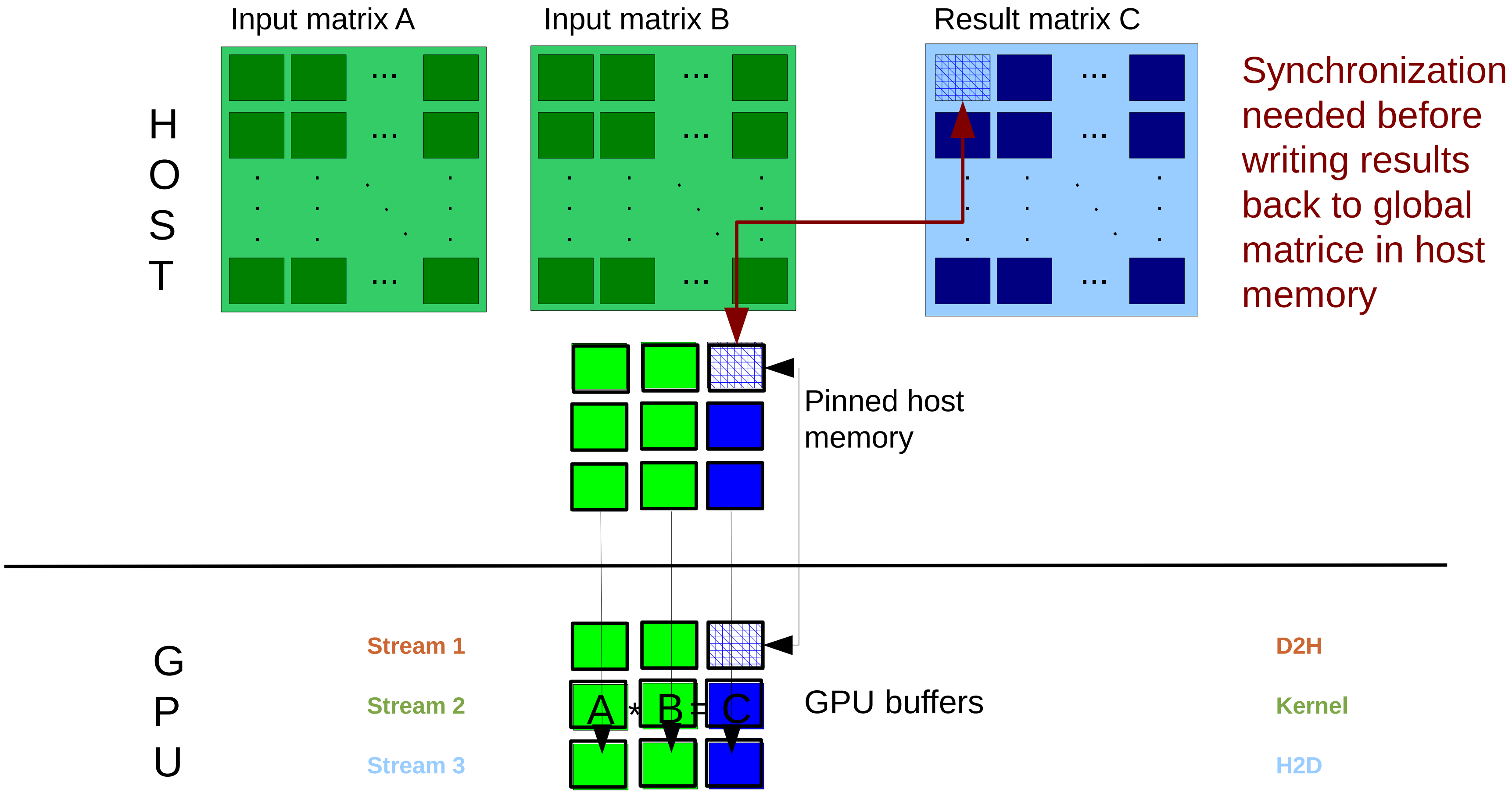
Tiled Matrix Multiplication – Using Streams

- Example: 3 streams



Tiled Matrix Multiplication – Using Streams

- Example: 3 streams



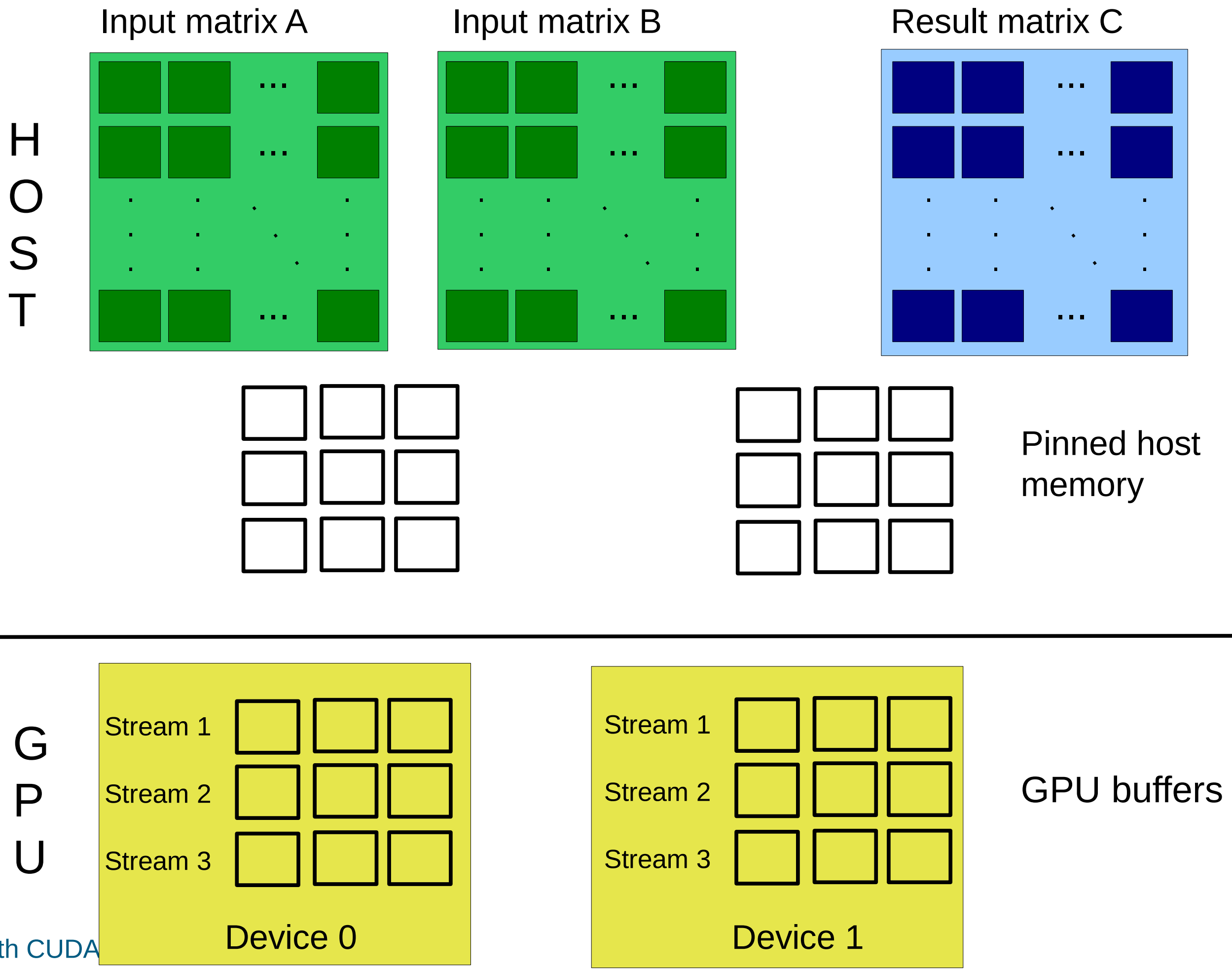
Exercise

Tiled Matrix Multiplication – Using Multi-GPUs with Streams

- Use all GPUs within a node (on Judge: 2)
- Each GPU uses several streams
  - *First fill all streams of a GPU then move to next GPU*

Tiled Matrix Multiplication – Using Multi-GPUs with Streams

- Example: 2 GPUs, 3 streams



Exercise