GPU PROGRAMMING WITH CUDA

Tiled Matrix Multiplication

25 – 29 April 2022 | Kaveh Haghighi Mood, Jochen Kreutz | JSC

OVERVIEW

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

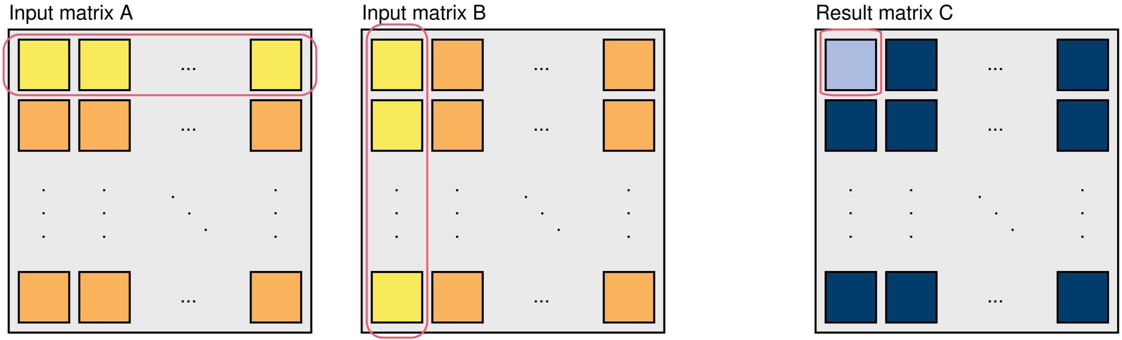
MOTIVATION

Block-wise Matrix Multiplication

- use cuBLAS library to get performance out of your GPU
  - easy to use
  - highly optimized
- what about using multiple GPUs within a node ?
- what about dealing with huge matrices that won’t entirely fit into GPU memory ?

TILED MATRIX MULTIPLICATION

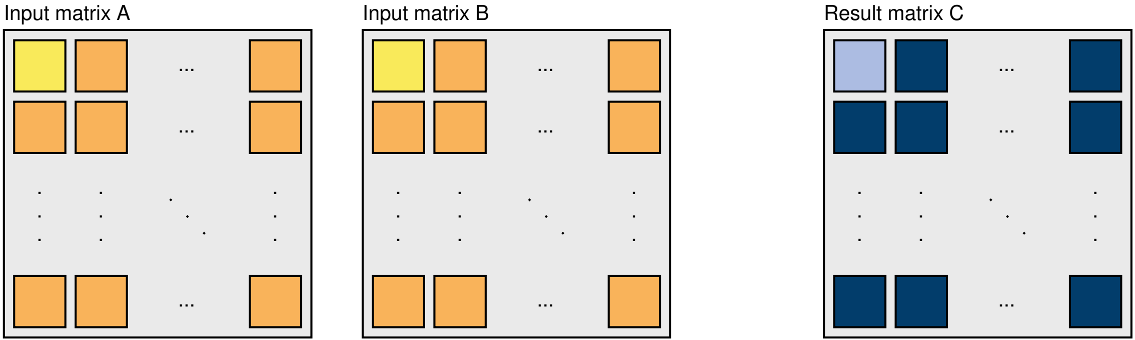
Block-wise Matrix Multiplication



- Split matrices into tiles (similar to block approach introduced with using shared memory)
- Allows for distribution of work onto different streams (and GPUs)
- Use highly optimized CuBLAS routines for block matrix multiplication

TILED MATRIX MULTIPLICATION

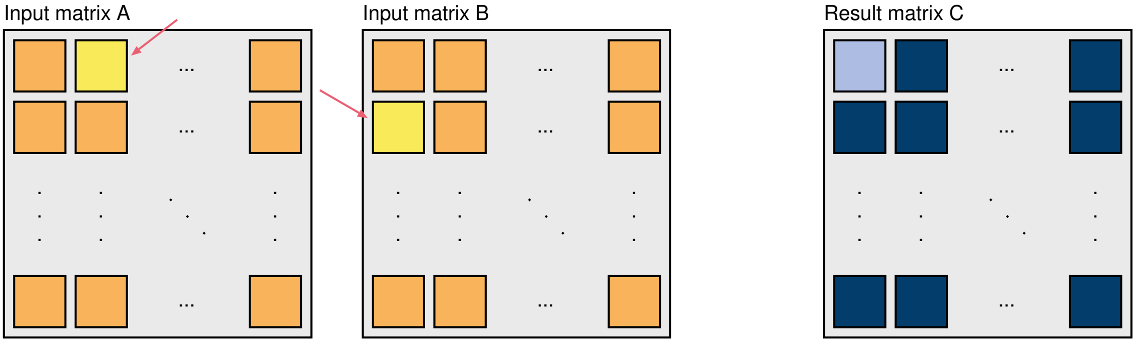
Block-wise Matrix Multiplication



- Do partial (block-wise) computation using CuBLAS library
- Sum up partial results

TILED MATRIX MULTIPLICATION

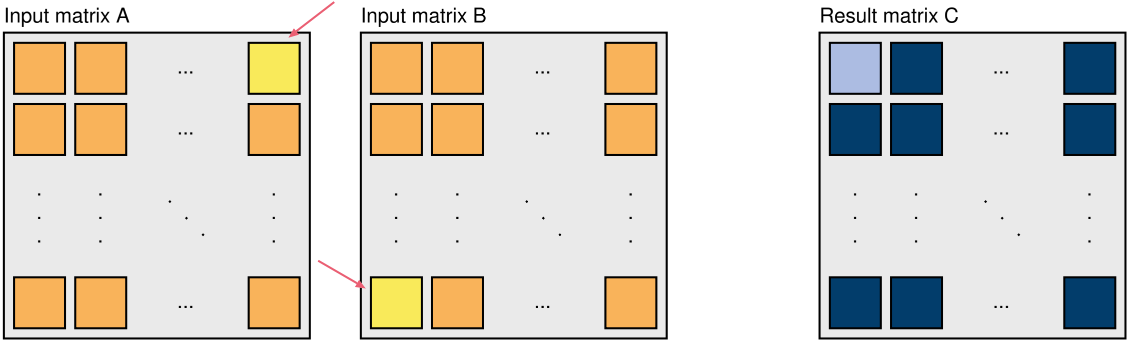
Block-wise Matrix Multiplication



- Do partial (block-wise) computation using CuBLAS library
- Sum up partial results

TILED MATRIX MULTIPLICATION

Block-wise Matrix Multiplication



- Do partial (block-wise) computation using CuBLAS library
- Sum up partial results

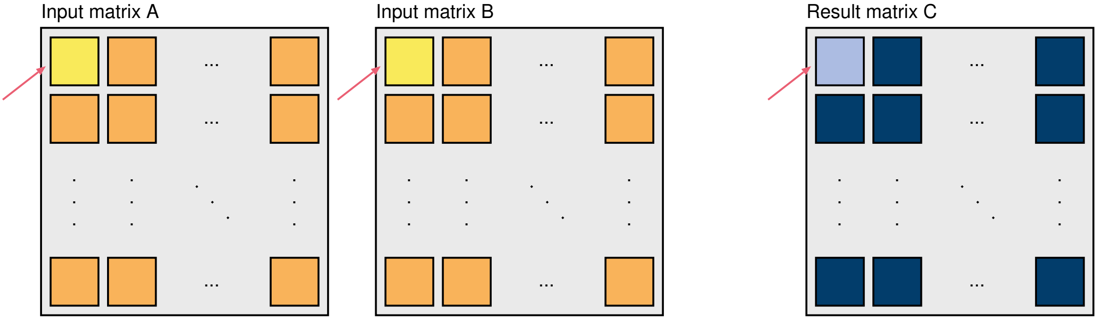
TILED MATRIX MULTIPLICATION

Block-wise Matrix Multiplication

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

TILED MATRIX MULTIPLICATION

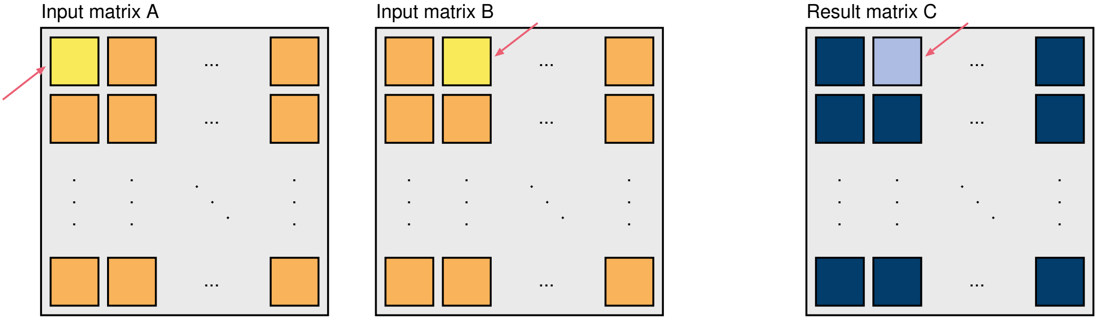
Block-wise Matrix Multiplication



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

TILED MATRIX MULTIPLICATION

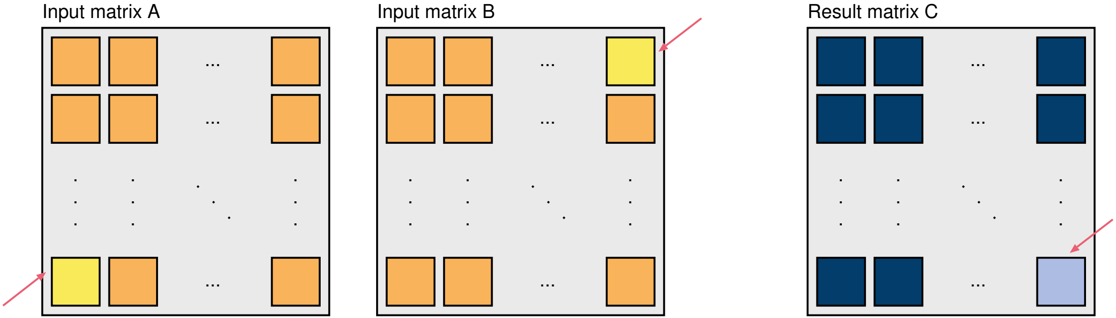
Block-wise Matrix Multiplication



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

TILED MATRIX MULTIPLICATION

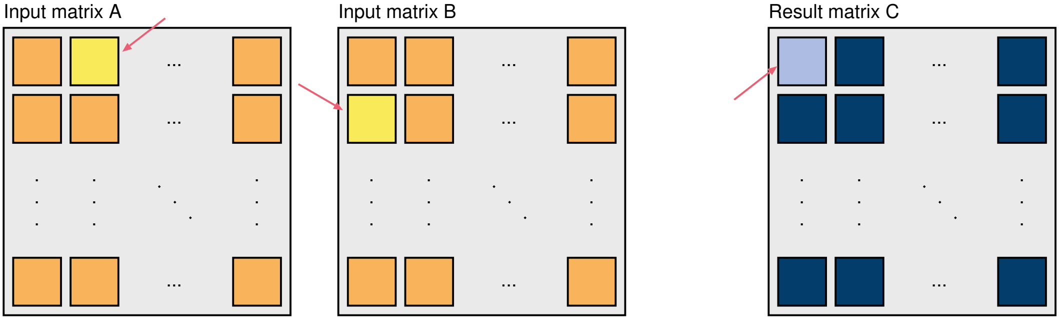
Block-wise Matrix Multiplication



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

TILED MATRIX MULTIPLICATION

Block-wise Matrix Multiplication



- Change order of computations and run over all tiles of the result matrix in the inner loop
- Do first computations for all tiles in the result matrix, 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 entirely fit into GPU memory
- 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 (sum up partial results)

DGEMM definition

```
C := alpha * A * B + beta * C
```

TILED MATRIX MULTIPLICATION

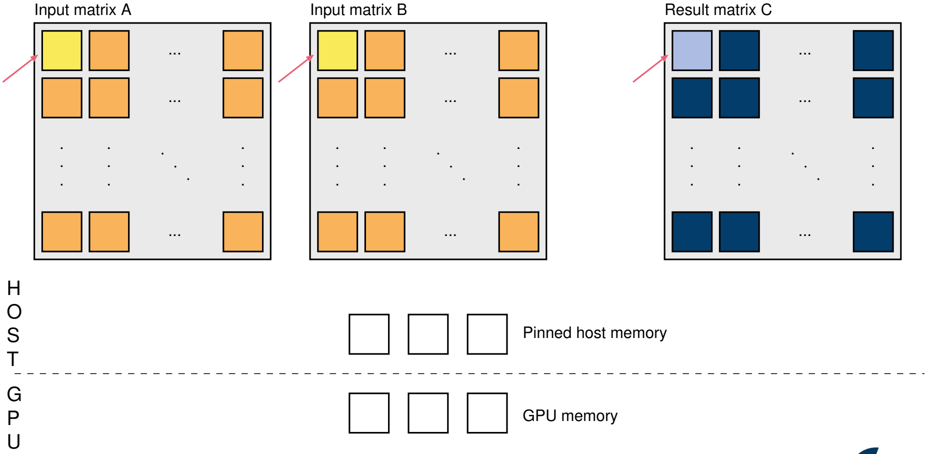
Implementation

Workflow:

- Init data (set elements of result matrix C to 0)
- Loop over tiles in C and in the input matrices to compute and sum up partial results
  1. Read input data (3 tiles) from global matrices (in host memory) to pinned buffers
  2. Transfer the three relevant tiles to the device
  3. Call `cublasDgemm` with `beta = 1`
  4. Read back partial results from device to pinned host buffer
  5. Write back partial result (1 tile) from pinned host buffer to global result matrix in host memory

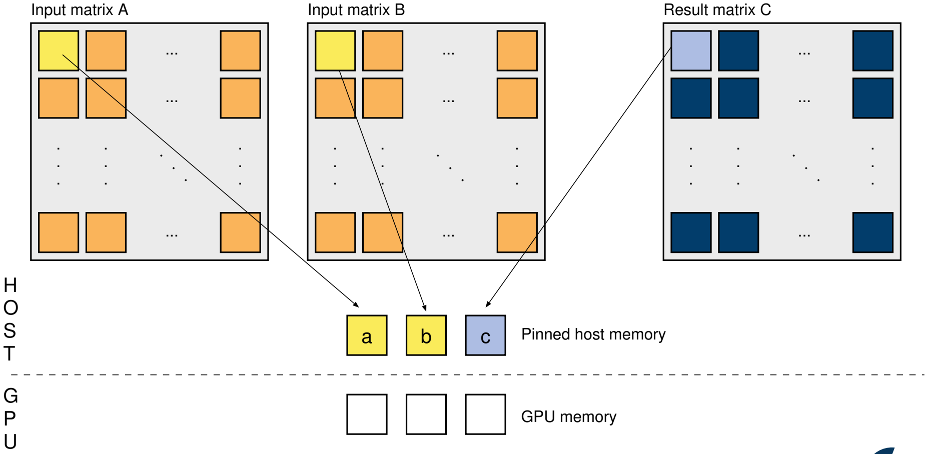
TILED MATRIX MULTIPLICATION

Memory buffers



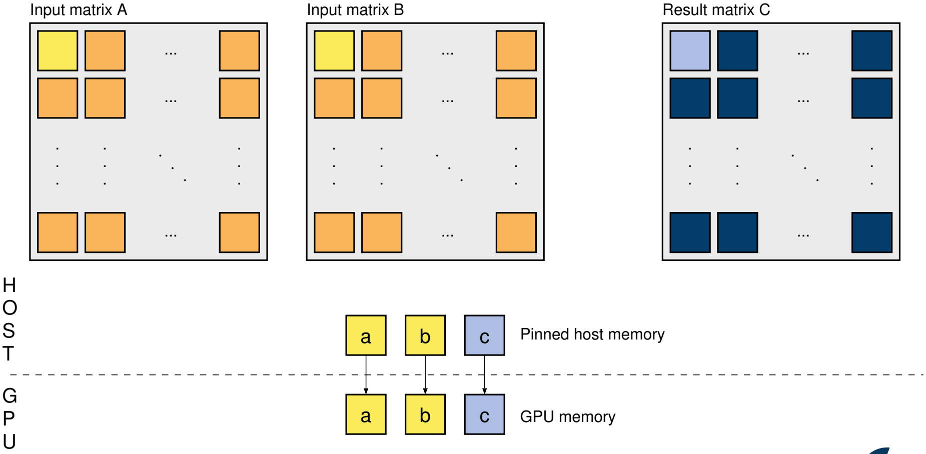
TILED MATRIX MULTIPLICATION

Memory buffers



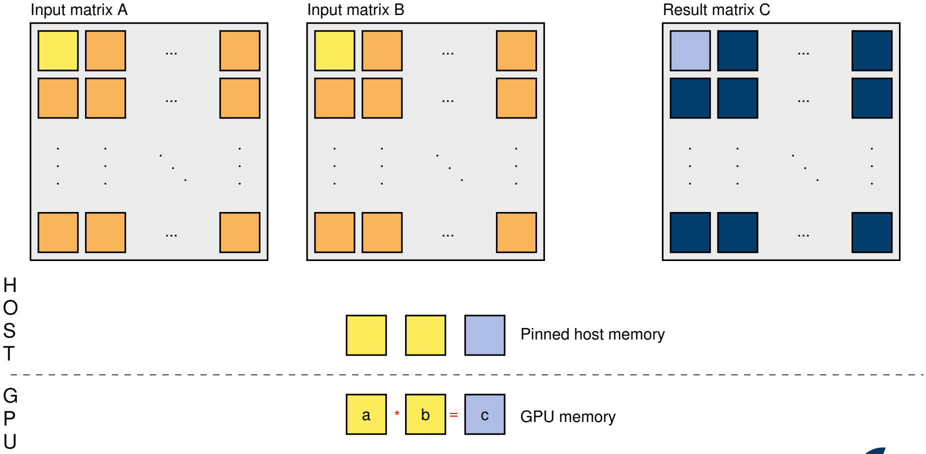
TILED MATRIX MULTIPLICATION

Memory buffers



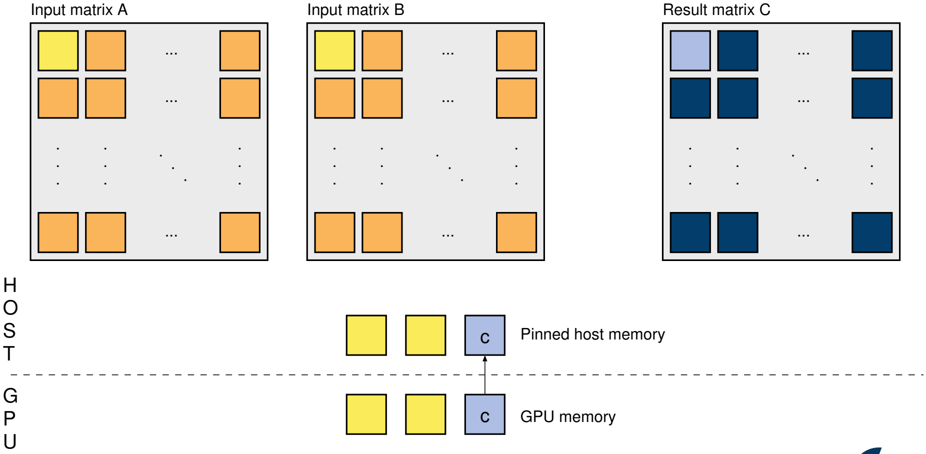
TILED MATRIX MULTIPLICATION

Memory buffers



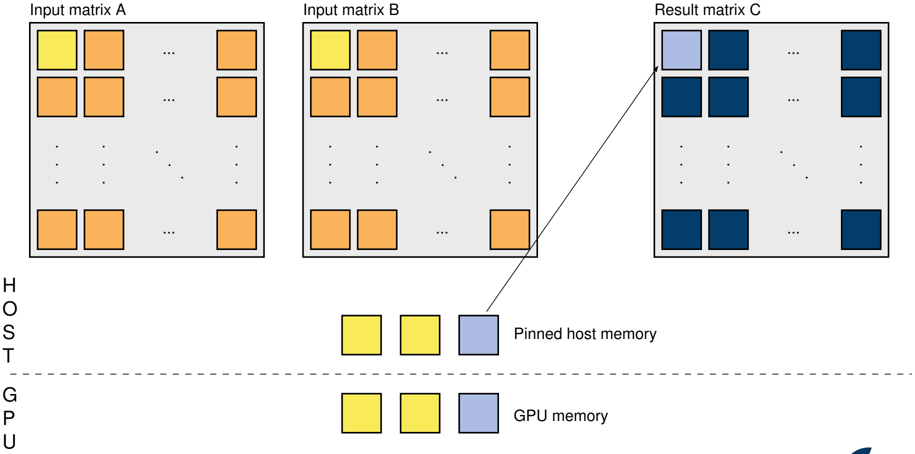
TILED MATRIX MULTIPLICATION

Memory buffers



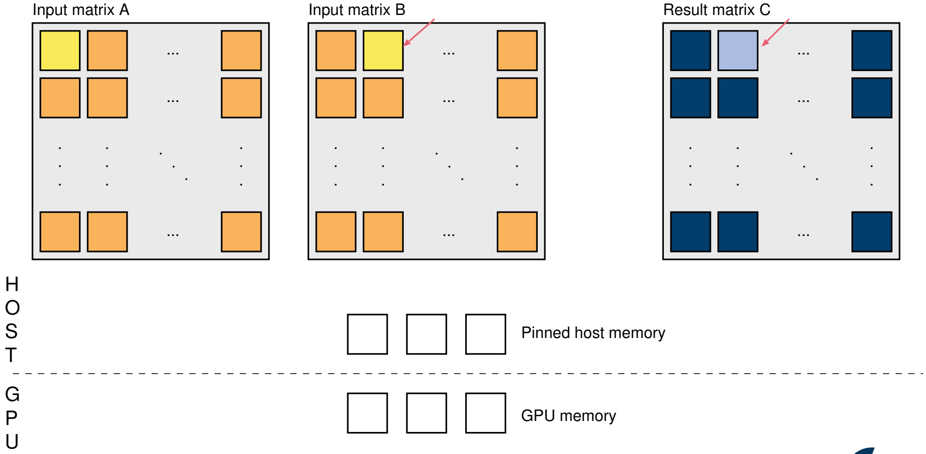
TILED MATRIX MULTIPLICATION

Memory buffers



TILED MATRIX MULTIPLICATION

Memory buffers



EXERCISE

Tiled Matrix Multiplication with Cuda

Location:

.../exercises/tasks/Instructions.ipynb

TILED MATRIX MULTIPLICATION

Using Streams

- Distribute computation of tiles to different streams
- Use asynchronous data transfers to overlap kernel executions and memory copies
- Each stream will use its own tile buffers (”multi-buffering”)
  - Simplifies implementation
  - Redundant data transfers can be hidden (by kernel execution)
- Synchronization is needed

Attention

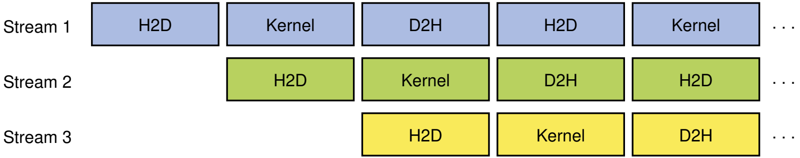
Two streams should not work on the same tile in C at the same time. Hence, number of streams has to be smaller than number of tiles in C !

TILED MATRIX MULTIPLICATION

Using Streams

Example: 3 streams

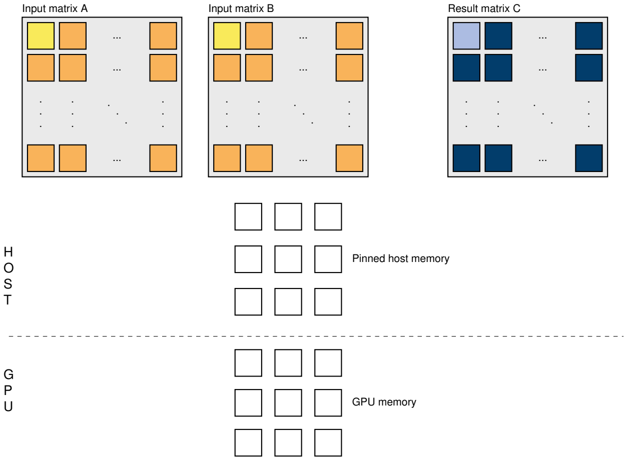
- For every tile:
  - H2D data transfer
  - Kernel execution (`cublasDgemm`)
  - D2H data transfer



Stream 1	H2D	Kernel	D2H	H2D	Kernel	...
Stream 2		H2D	Kernel	D2H	H2D	...
Stream 3			H2D	Kernel	D2H	...

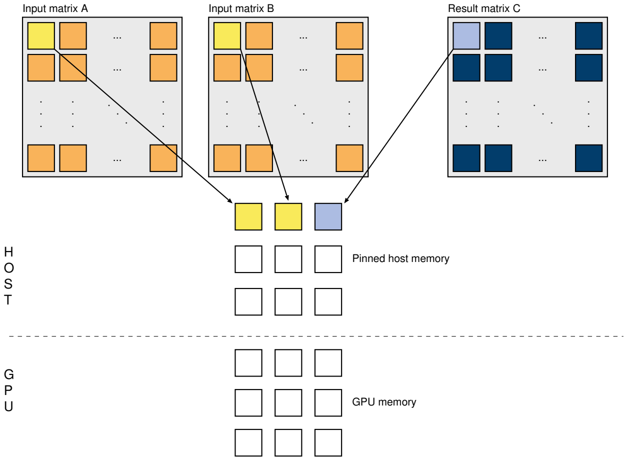
TILED MATRIX MULTIPLICATION

Using Streams



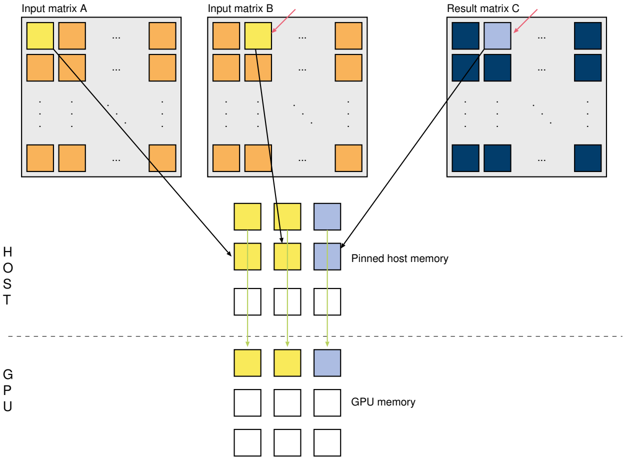
TILED MATRIX MULTIPLICATION

Using Streams



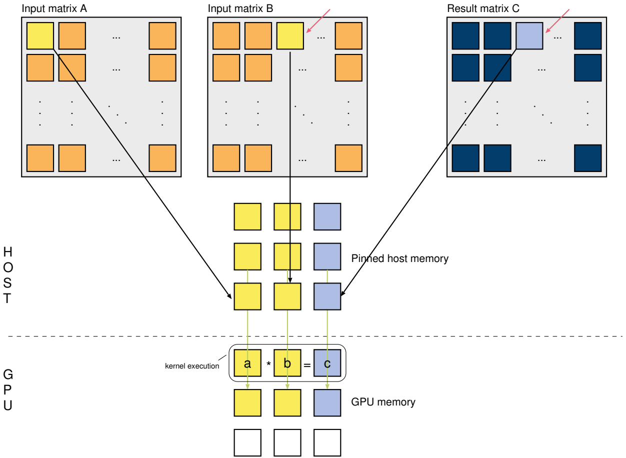
TILED MATRIX MULTIPLICATION

Using Streams



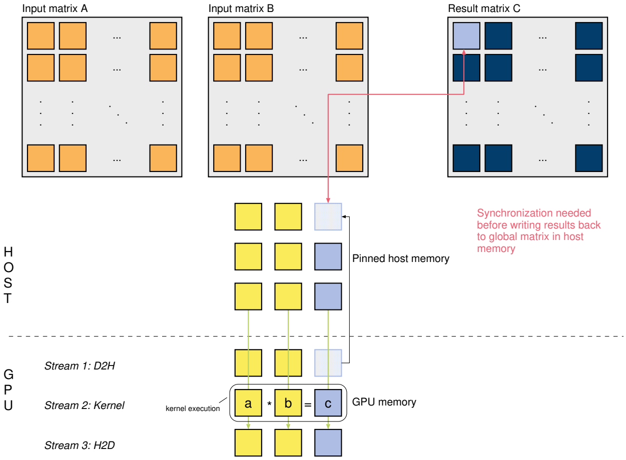
TILED MATRIX MULTIPLICATION

Using Streams



TILED MATRIX MULTIPLICATION

Using Streams



Synchronization needed before writing results back to global matrix in host memory

EXERCISE

Tiled Matrix Multiplication with Cuda using Streams

Location:

.../exercises/tasks/Instructions.ipynb

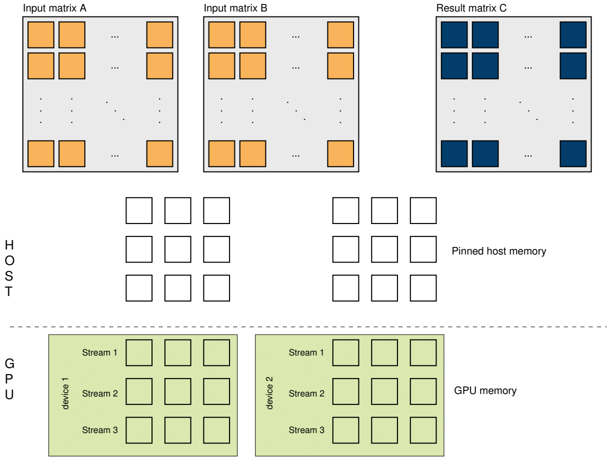
TILED MATRIX MULTIPLICATION

Using Streams on multiple GPUs

- Use all GPUs within a node
- Each GPU uses several streams
    - Fill all streams of a GPU first, then move on to next GPU

TILED MATRIX MULTIPLICATION

Using Streams on multiple (e.g. 2) GPUs



MULTI-GPU LIBRARIES

Extensions of CuBLAS library

- The cuBLASXt API of cuBLAS exposes a multi-GPU capable Host interface
  - no restriction on the sizes of the matrices as long as they can fit into the host memory
  - offers the possibility to offload some of the computation to the host CPU
- cuBLASMg provides a state-of-the-art multi-GPU matrix-matrix multiplication
  - Currently a part of the CUDA Math Library Early Access Program

EXERCISE

Tiled Matrix Multiplication with Cuda using Streams on multiple GPUs

Location:

.../exercises/tasks/Instructions.ipynb