CINECA

CUDA streams

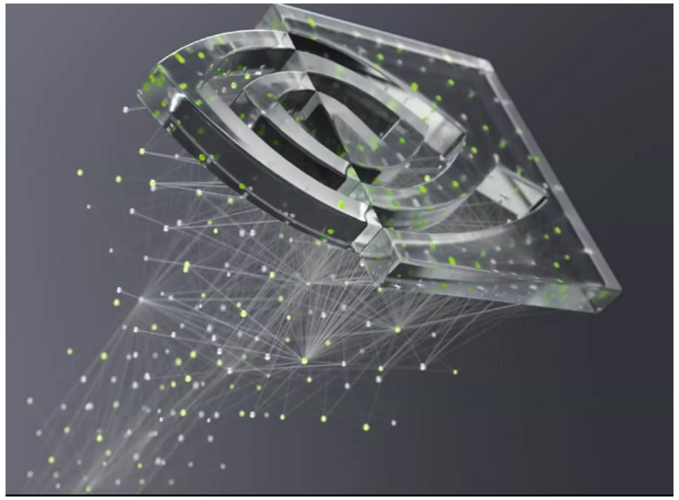
Lara Querciagrossa, Andrew Emerson, Nitin Shukla, Luca Ferraro, Sergio Orlandini

l.querciagrossa@cineca.it

July 12th, 2022

In this lecture…

- ✓ Streams
- ✓ Streams behavior
- ✓ Using streams



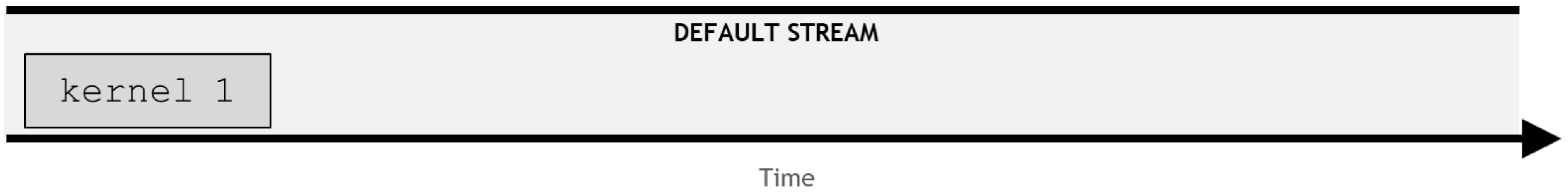
Streams

Streams

- A **stream** is a **series of commands** (kernel execution and memory transfer) **that execute in order**.

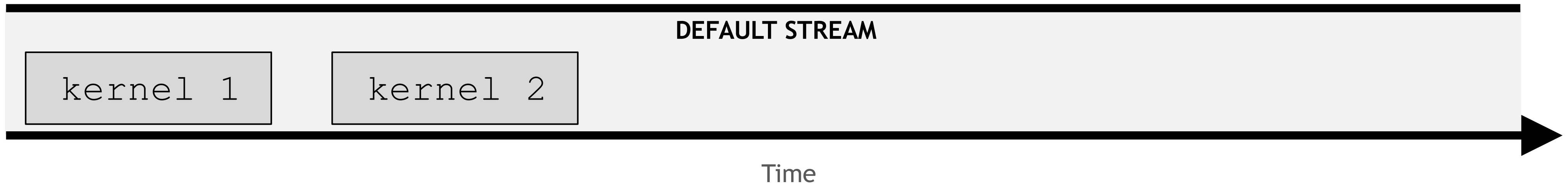
- If no explicit CUDA stream has been specified, CUDA kernels are executed in a **default stream**.

- **Non-default CUDA streams** can be used to perform multiple operation **concurrently** in different streams.



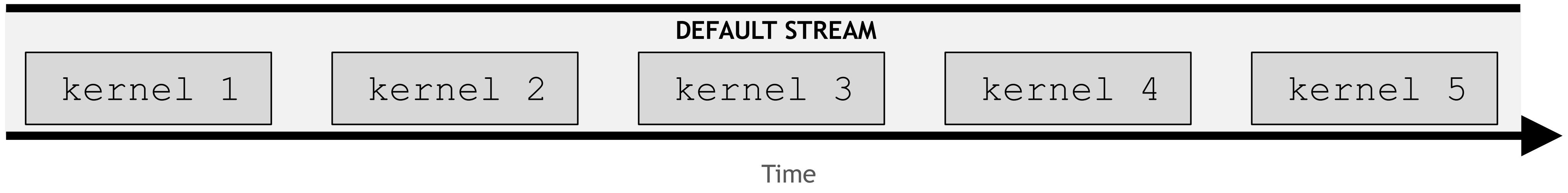
Streams

- **In any stream**, both default and not, **instructions execute in order**: an instruction must be completed before the next one can begin.



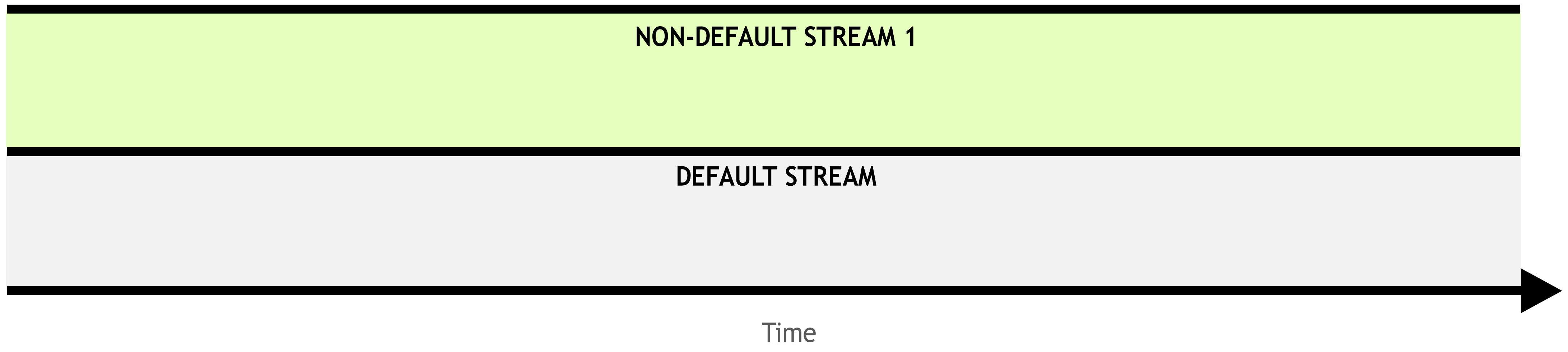
Streams

- **In any stream**, both default and not, **instructions execute in order**: an instruction must be completed before the next one can begin.



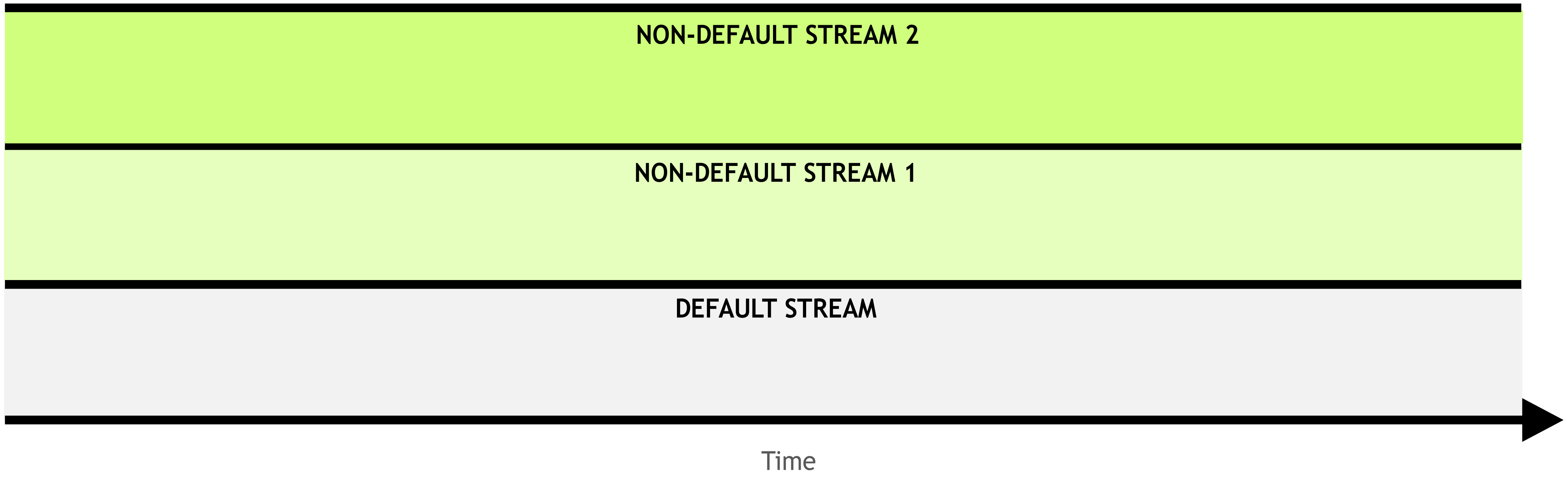
Streams

- Additional **non-default streams** can be created for kernel execution.



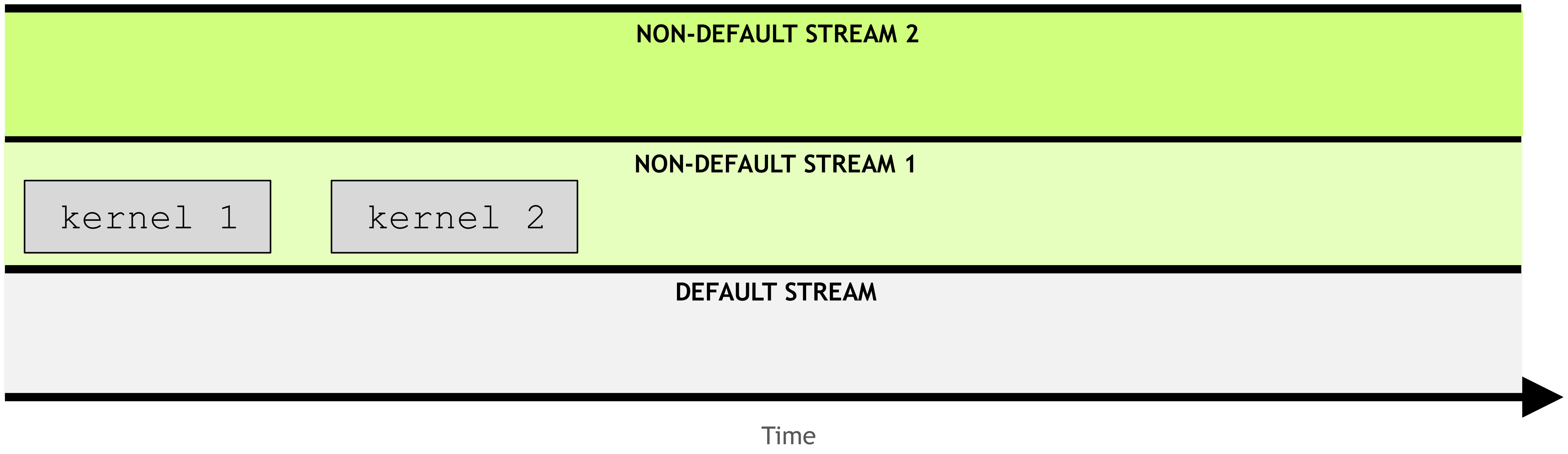
Streams

- Additional **non-default streams** can be created for kernel execution.



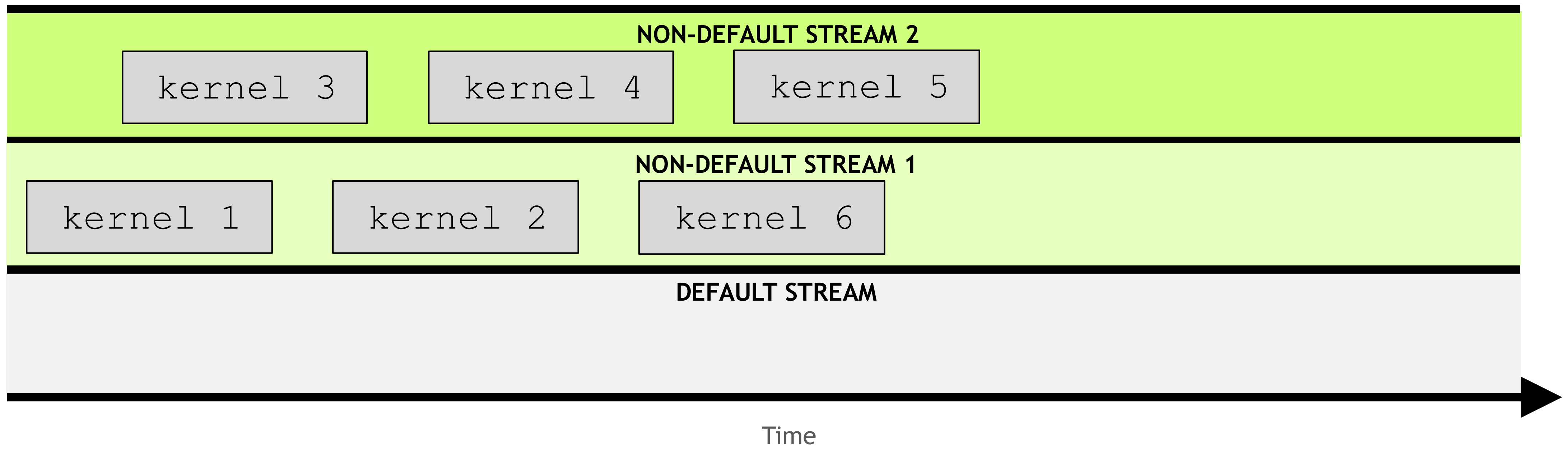
Streams

- Kernels in non-default stream must execute in order as well.



Streams

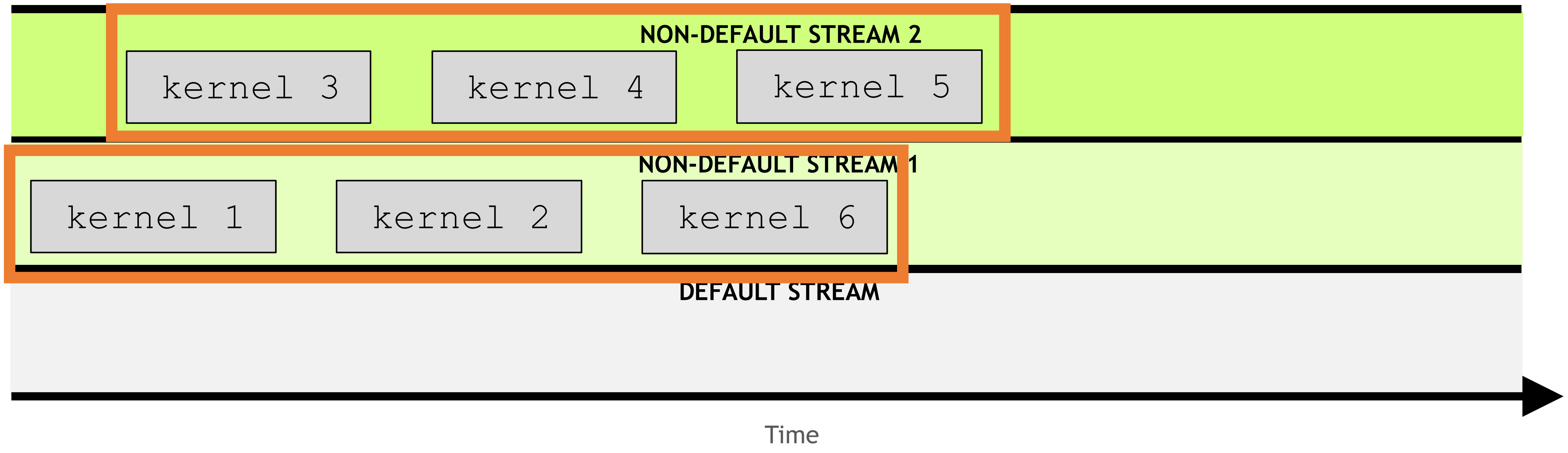
- But! Kernels in **different non-default streams can interact concurrently**.



Streams behavior

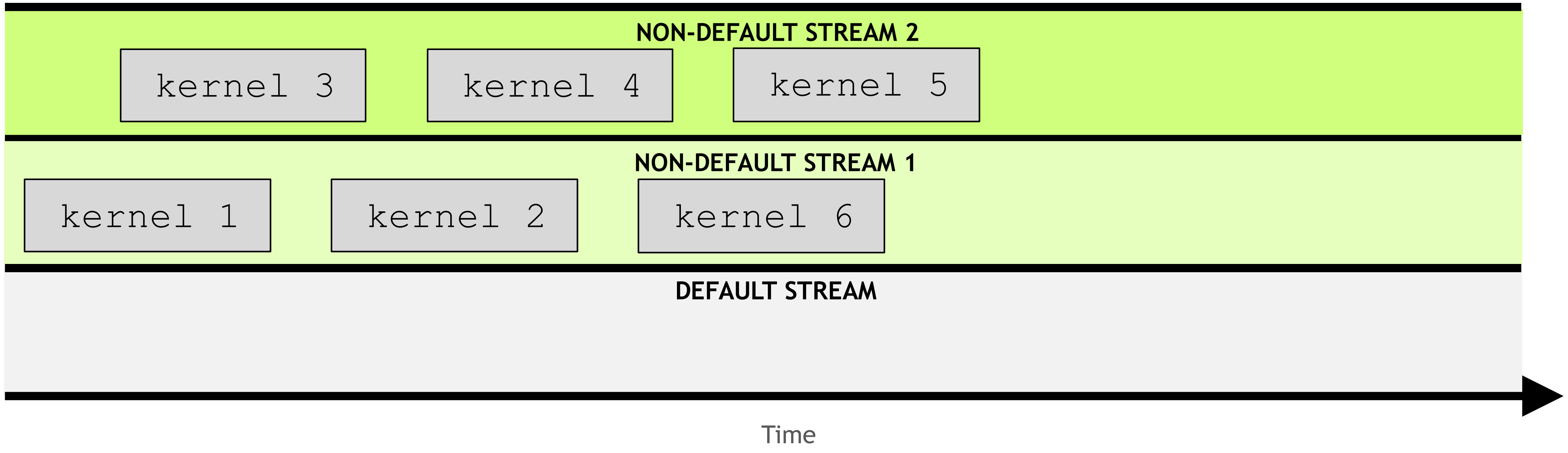
Rules governing the behavior of streams

1. Operations within a given stream occurs in order.



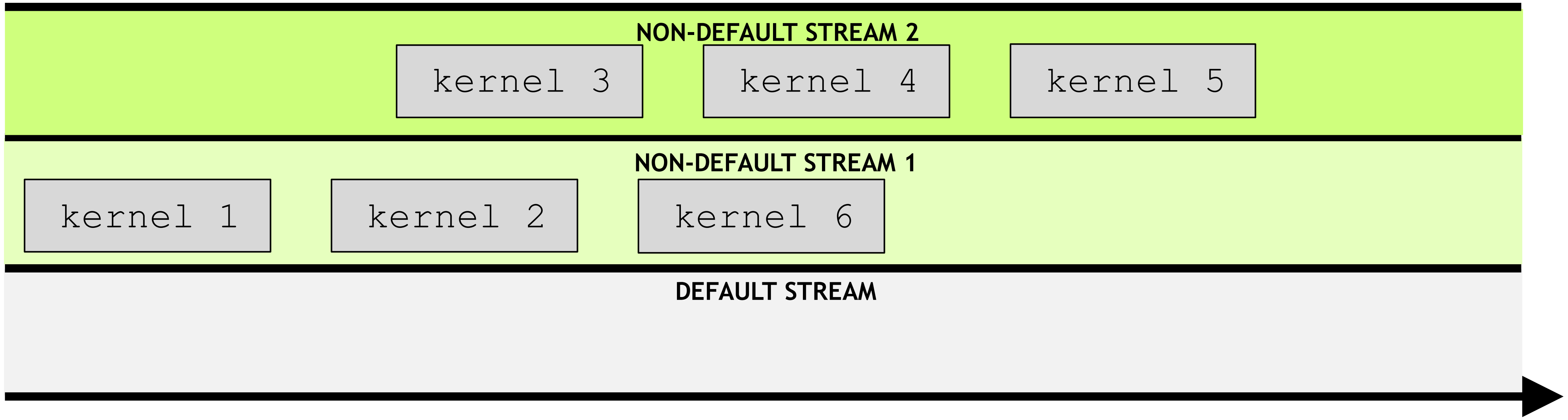
Rules governing the behavior of streams

1. Operations within a given stream occurs in order.
2. **Operations in different non-default streams are not guaranteed to operate in any specific order relative to each other.**



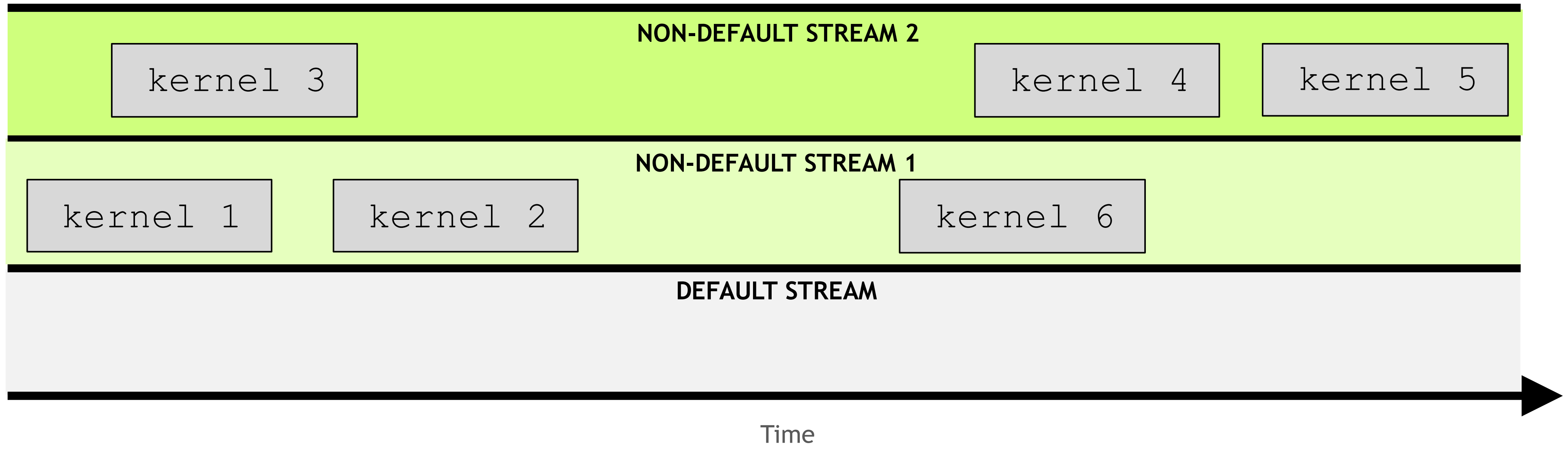
Rules governing the behavior of streams

1. Operations within a given stream occurs in order.
2. **Operations in different non-default streams are not guaranteed to operate in any specific order relative to each other.**



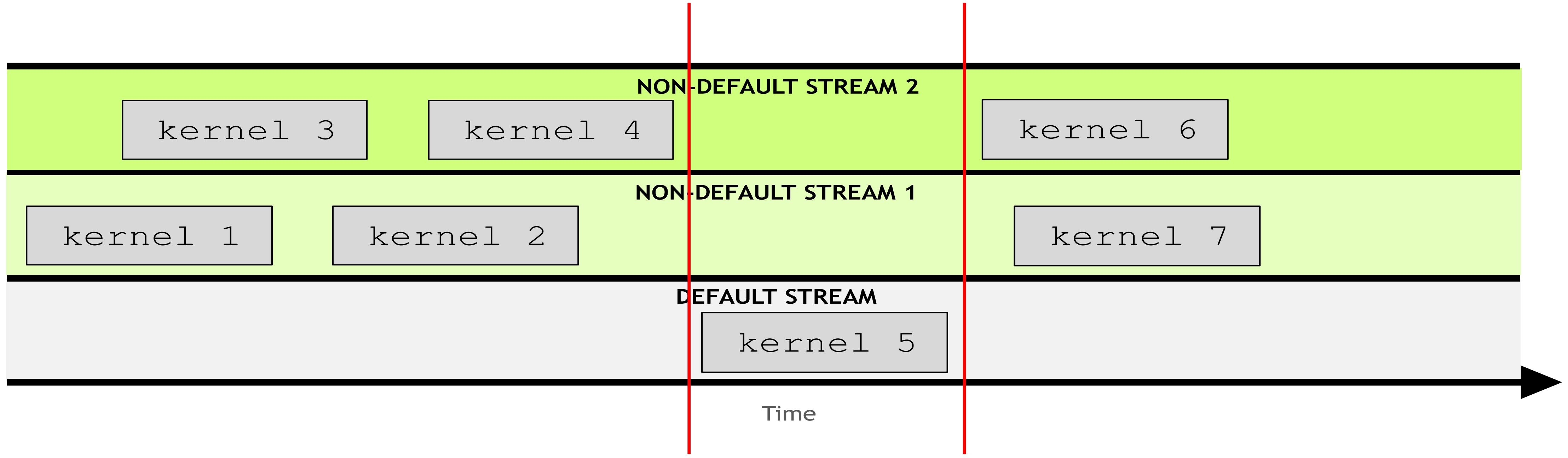
Rules governing the behavior of streams

1. Operations within a given stream occurs in order.
2. **Operations in different non-default streams are not guaranteed to operate in any specific order relative to each other.**



Rules governing the behavior of streams

1. Operations within a given stream occurs in order.
2. Operations in different non-default streams are not guaranteed to operate in any specific order relative to each other.
3. **The default stream is blocking and will both wait for all other streams to complete before running, and, will block other streams from running until it completes.**



Using streams

Creating, using and destroying non-default streams

- CUDA streams should be created as follows:

```
cudaStream_t stream;
cudaStreamCreate(&stream);
```

reference

- To launch a CUDA kernel in a non-default CUDA stream a 4th argument should be passed to the execution configuration:

```
someKernel<<<number_of_blocks, threads_per_block, 0, stream>>>();
```

Off-topic: number of bytes of **shared memory** (small and fast memory mounted on each SM) to be dynamically allocated per block.

- CUDA non-default streams should be destroyed when not used:

```
cudaStreamDestroy(stream);
```

value

Exercise: 19_print_number_streams.cu

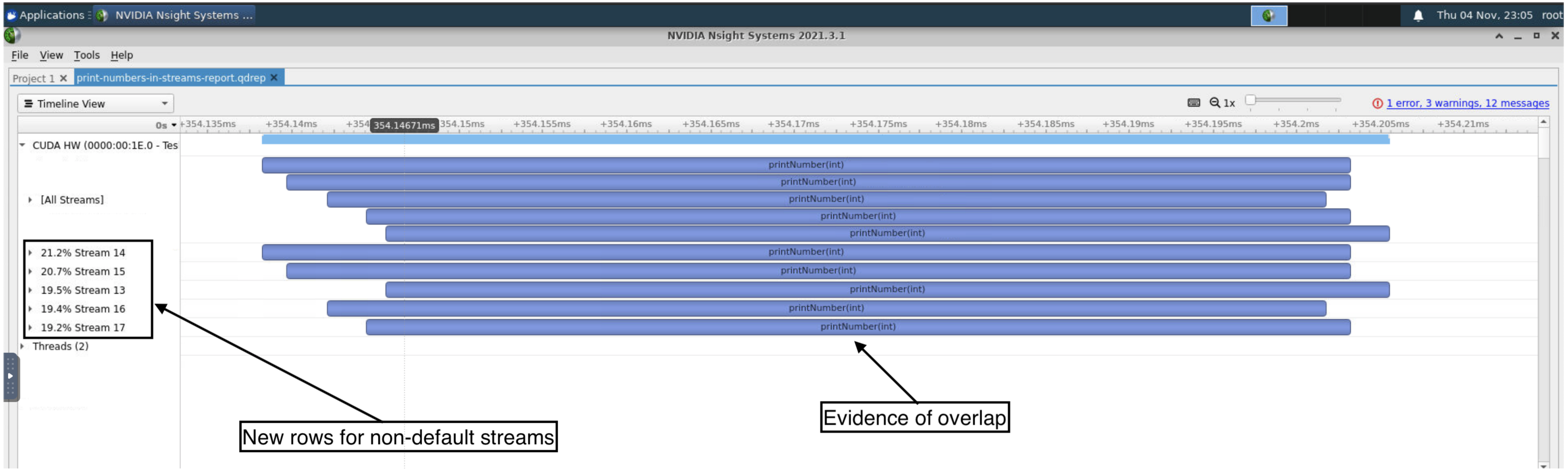
This exercise starts from a very simple `printNumber` kernel which accepts an integer and prints it. The kernel is being executed 5 times, using a for-loop, and passing each launch the number of the for-loop's iteration. These iterations run serially since they are all in the default stream.

Refactor the code so that each kernel launch occurs in its own non-default streams.

Will kernels now run in parallel?

Exercise: 19_print_number_streams.cu

In the next days you will learn how to inspect the profile with Nsight System.



Exercise: 20_vector_add_streams.cu

The starting point of this exercise is vector addition application you have been working on in exercises 11, 15 and 17.
Currently, it launches an initialization kernel 3 times, once for each of the 3 vectors needed in the `vectorAdd` kernel.

Refactor it to **launch each of the 3 initialization kernel launches in their own non-default stream.**

Be sure to still see the success message print when compiling and running your solution.

In the next days you will learn how to inspect the profile with Nsight System.

References

References

- Previous editions of this school at CINECA
- Oakridge National Laboratory's "Introduction to CUDA C++": https://www.olcf.ornl.gov/calendar/introduction-to-cuda-c/
- NVIDIA DL Institute Online Course: **main source of exercises**
- www.computerhope.com/jargon/p/pagefaul.htm
- blogs.nvidia.com
- Wikipedia

THANK YOU!

Lara Querciagrossa
l.querciagrossa@cineca.it