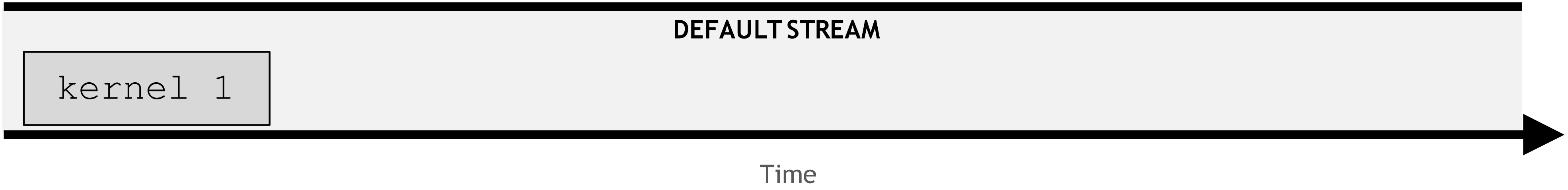
Concurrent CUDA Streams

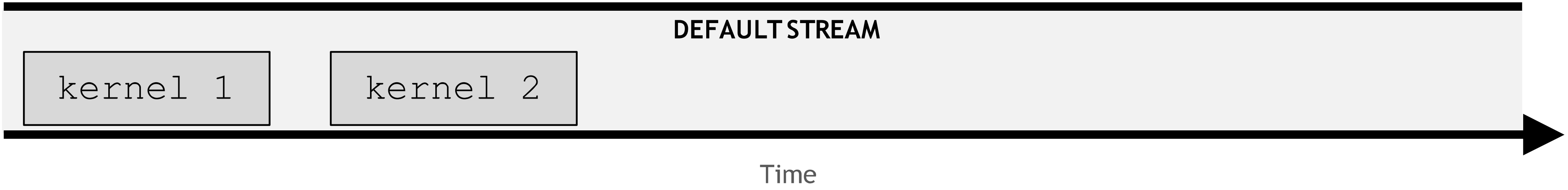
A **stream** is a series of instructions, and CUDA has a **default stream**

DEFAULT STREAM
Time

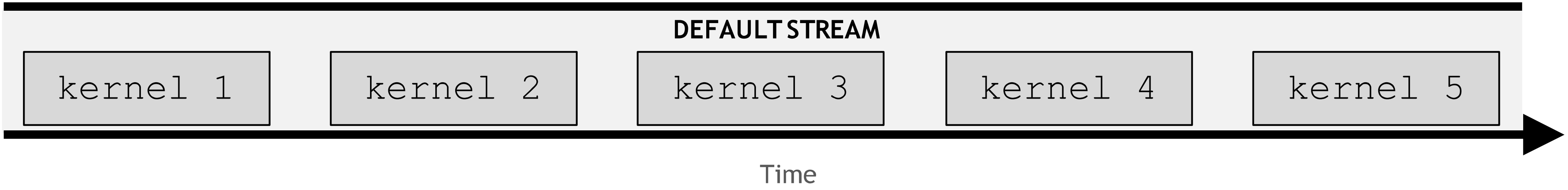
By default, CUDA kernels run in the **default stream**



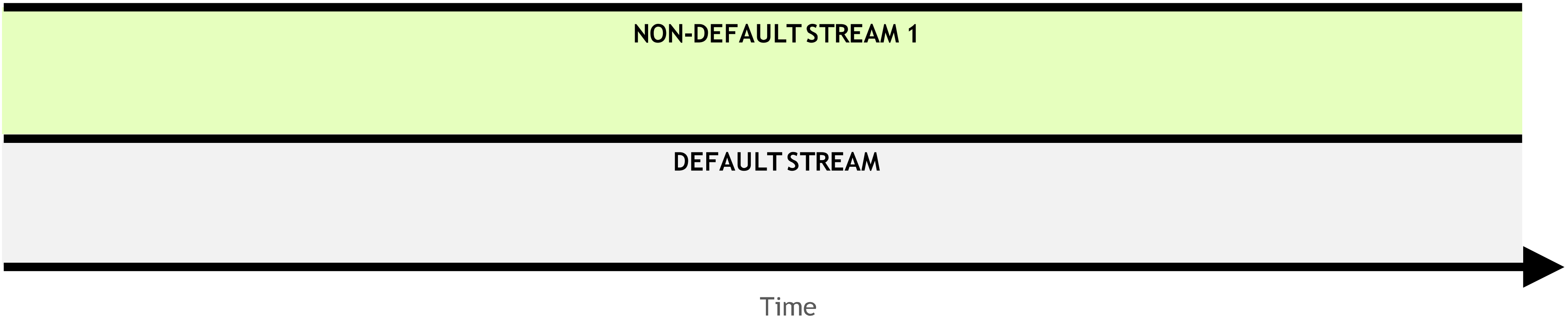
In any stream, including the default, an instruction in it (here a kernel launch) must complete before the next can begin



In any stream, including the default, an instruction in it (here a kernel launch) must complete before the next can begin



Non-default streams can also be created for kernel execution

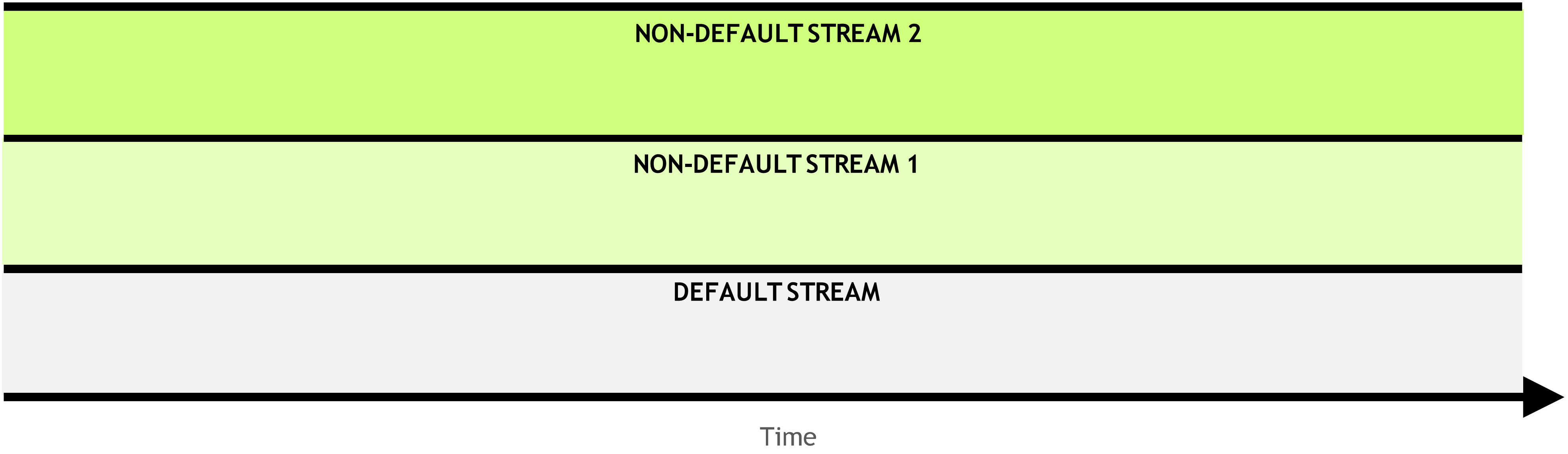


NON-DEFAULT STREAM 1

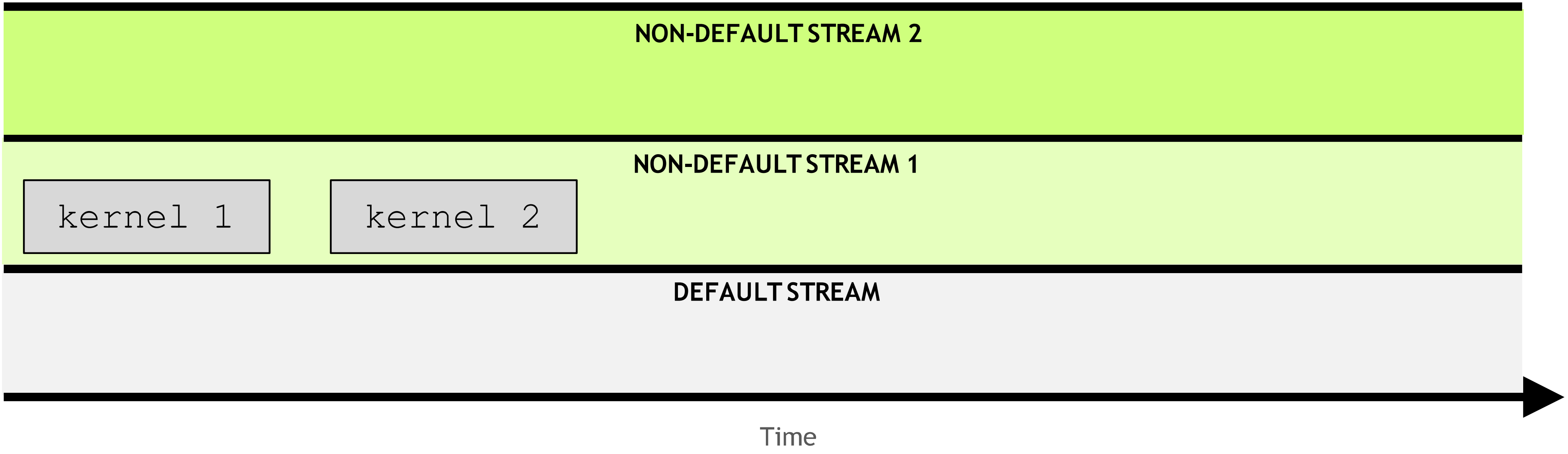
DEFAULT STREAM

Time

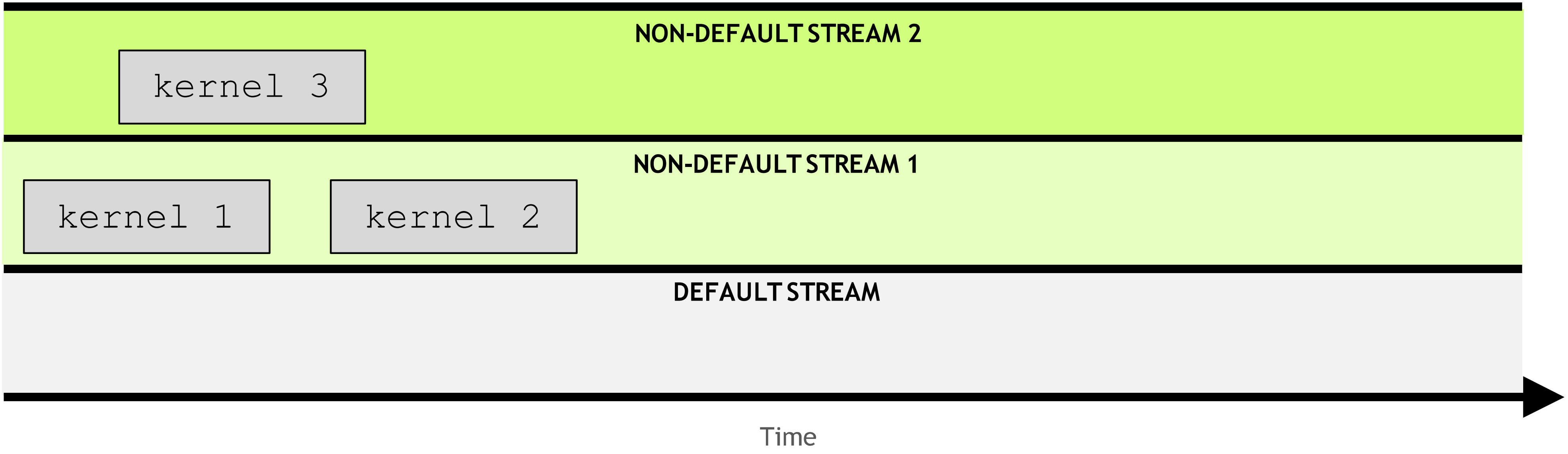
Non-default streams can also be created for kernel execution



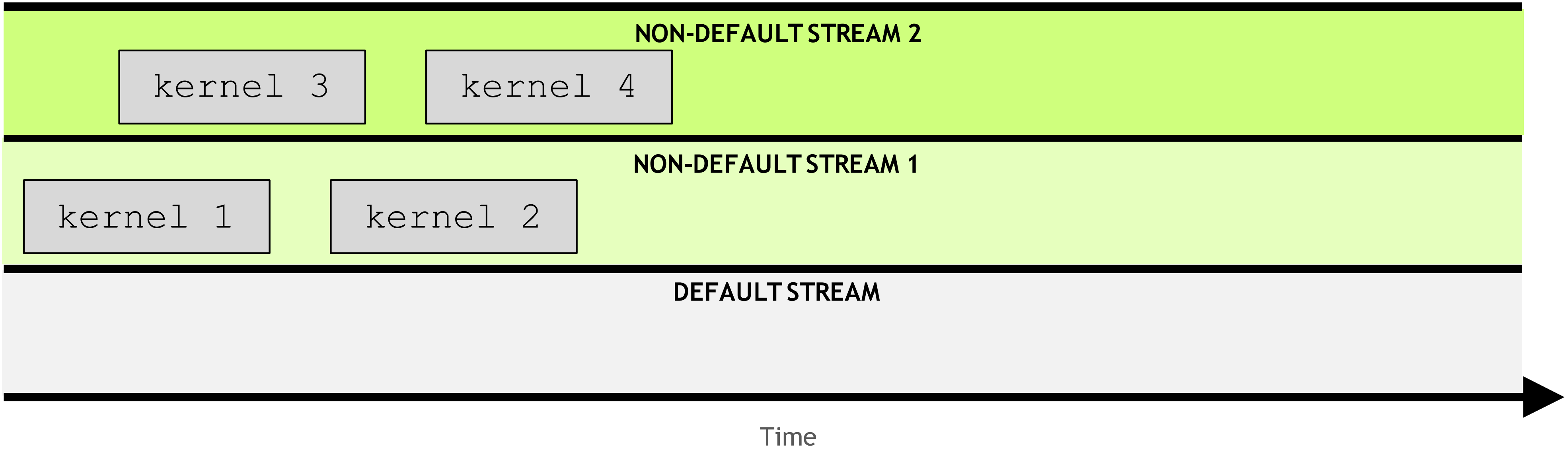
Kernels within any single stream must execute in order



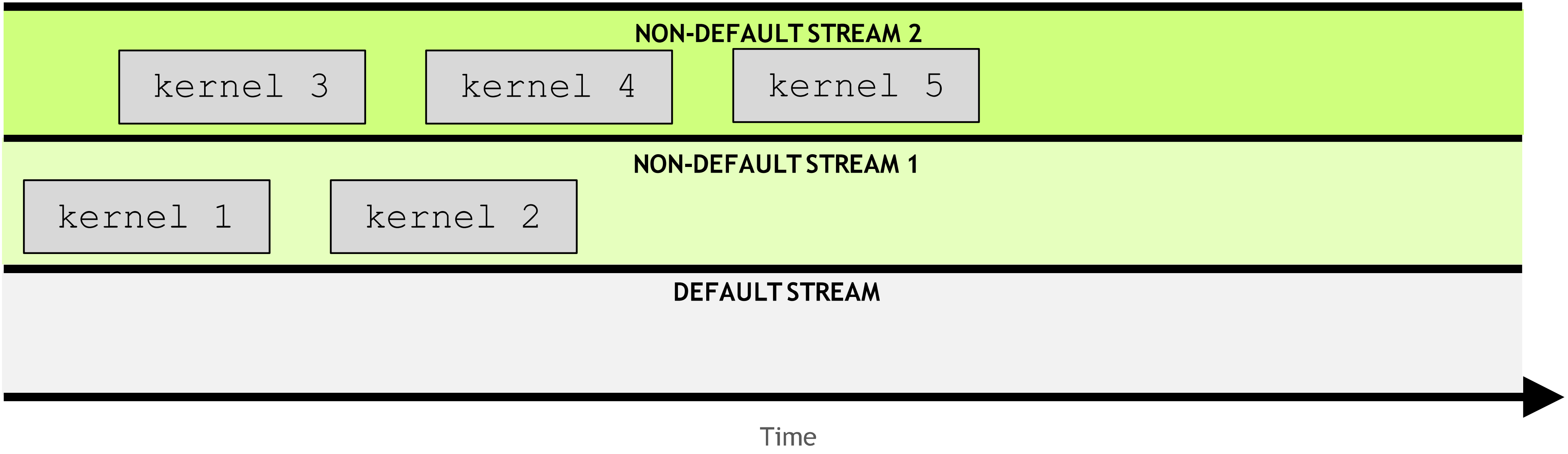
However, kernels in **different, non-default streams**, can interact concurrently



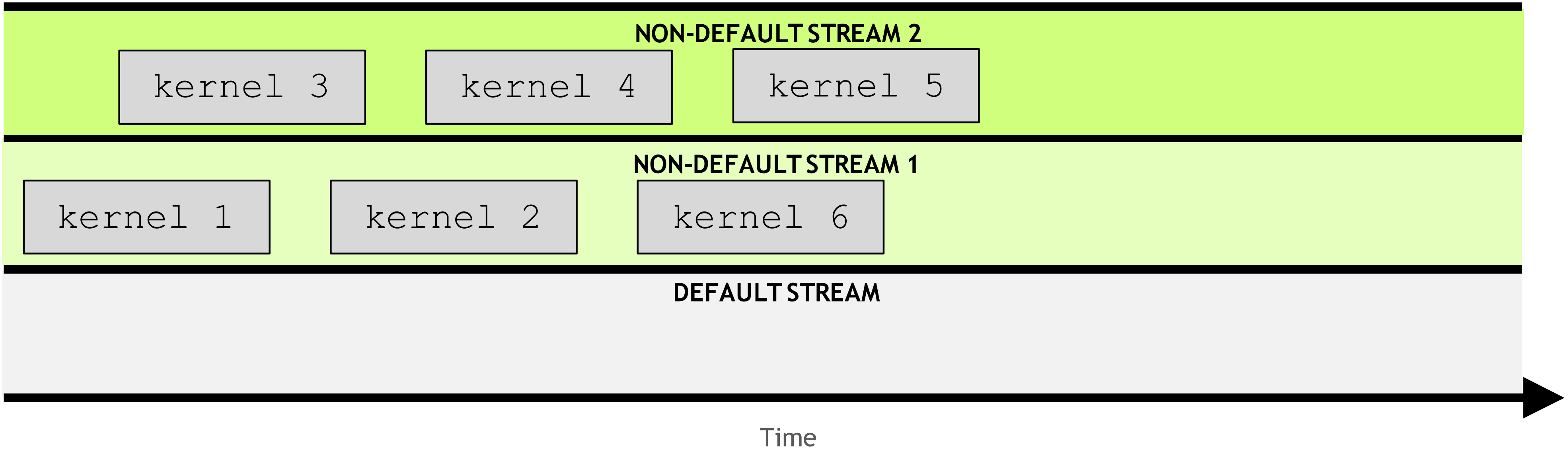
However, kernels in **different, non-default streams**, can interact concurrently



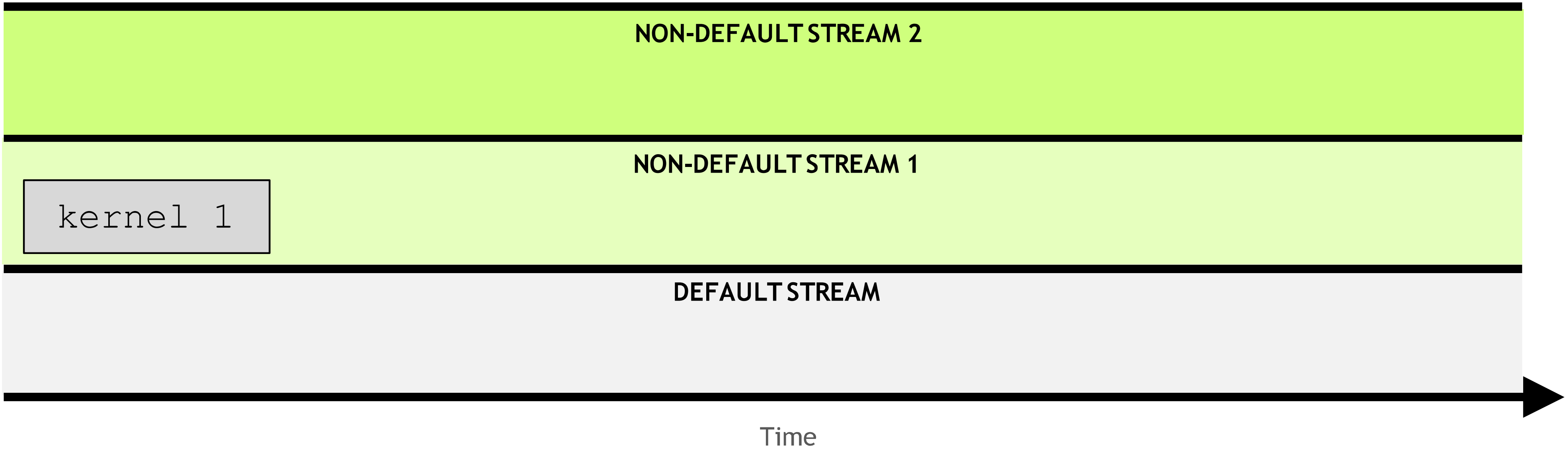
However, kernels in **different, non-default streams**, can interact concurrently



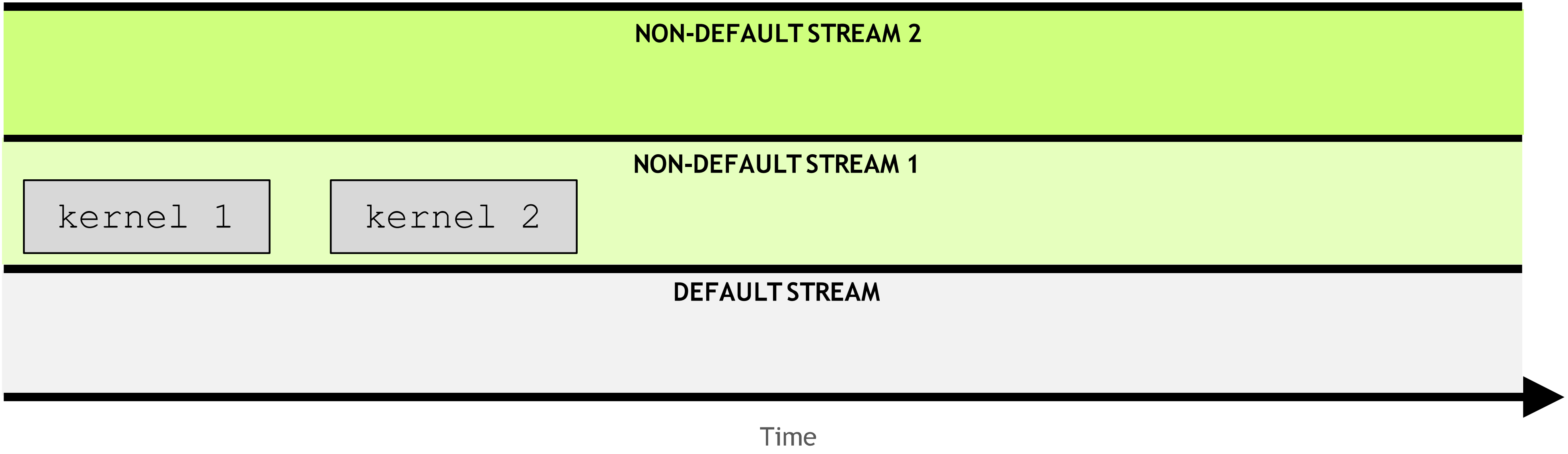
However, kernels in **different, non-default streams**, can interact concurrently



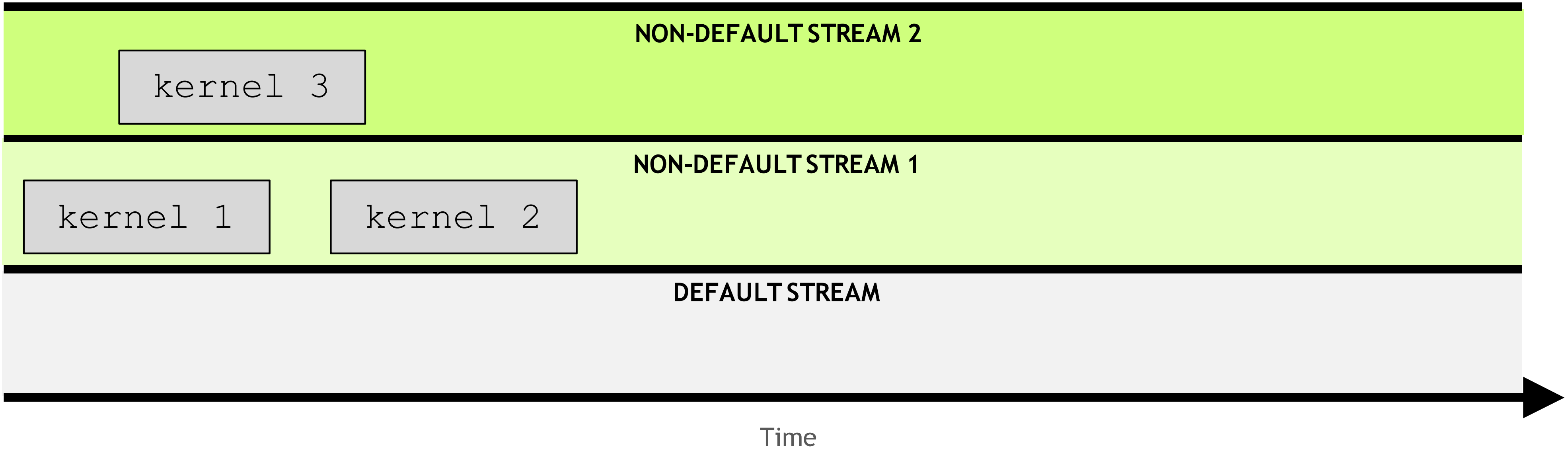
The default stream is special: **it blocks all kernels in all other streams**



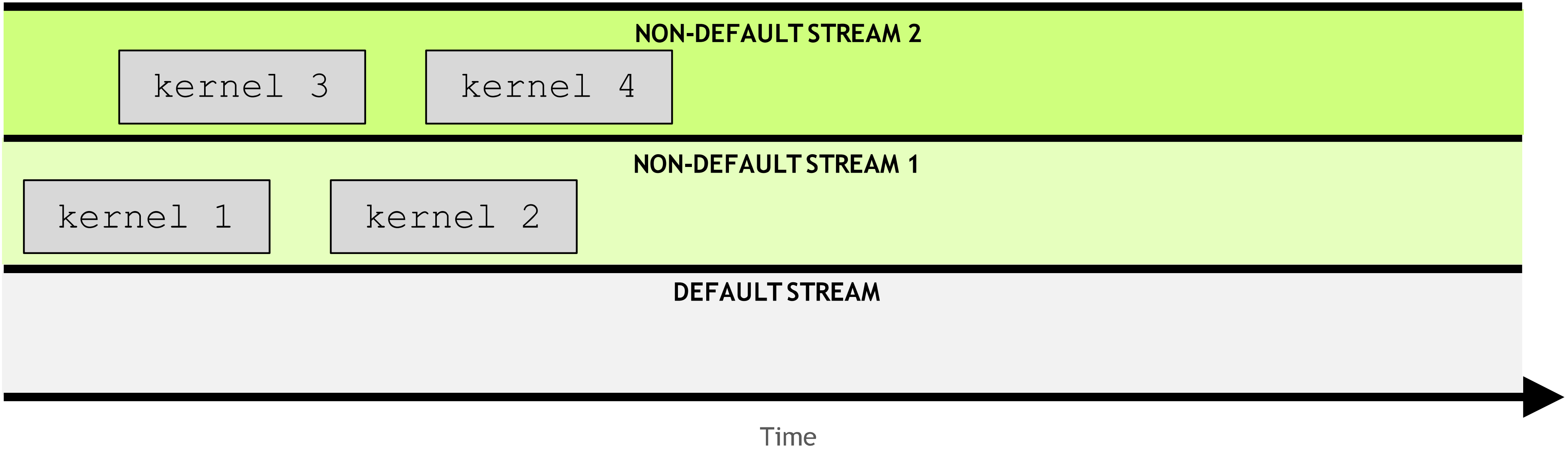
The default stream is special: **it blocks all kernels in all other streams**



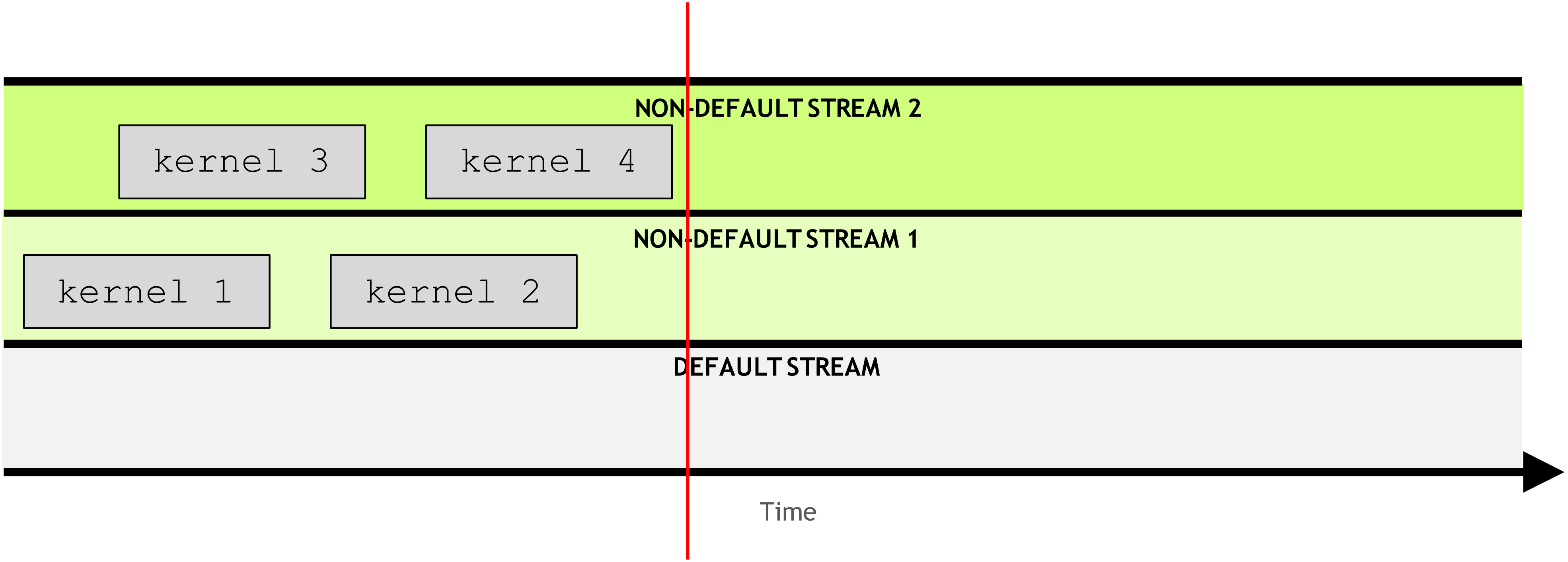
The default stream is special: **it blocks all kernels in all other streams**



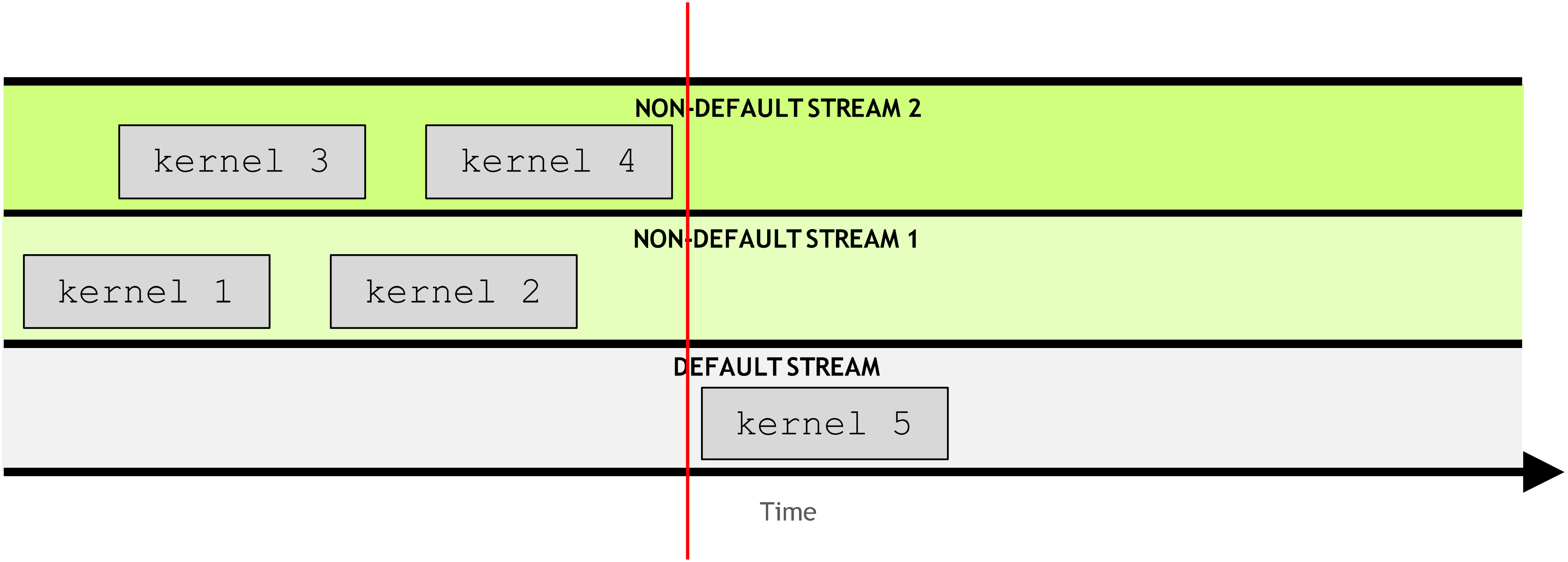
The default stream is special: **it blocks all kernels in all other streams**



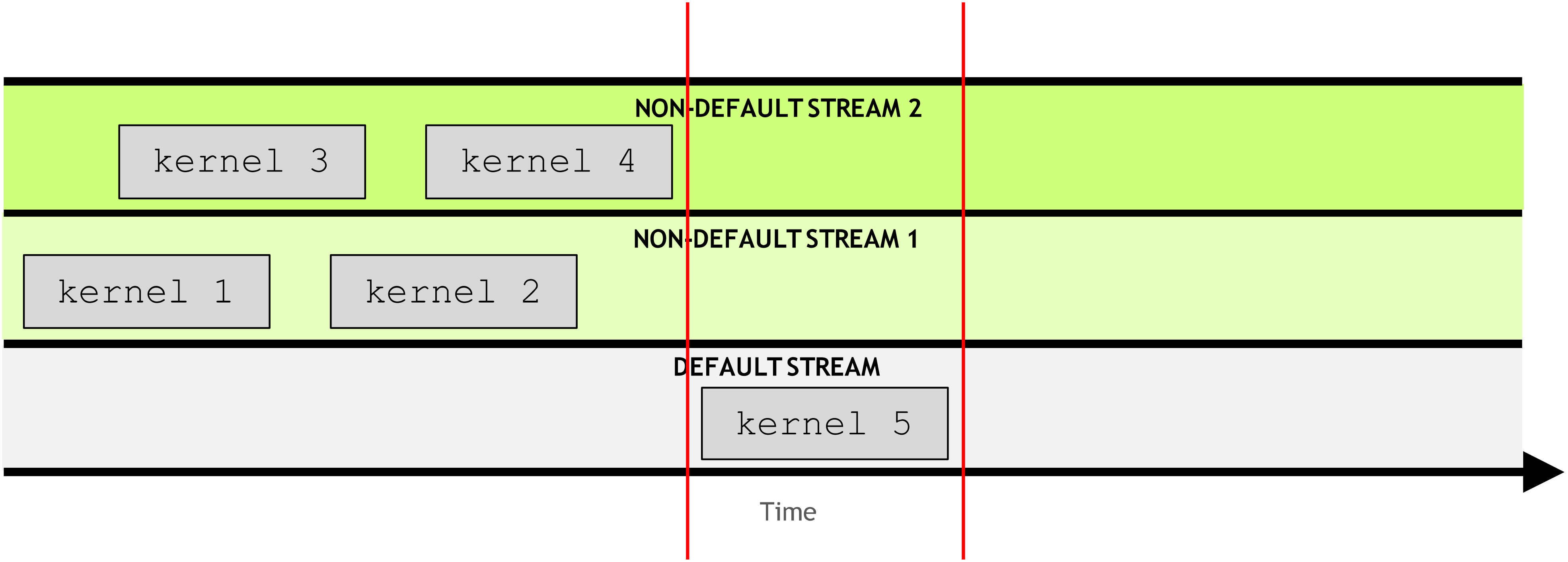
The default stream is special: **it blocks all kernels in all other streams**



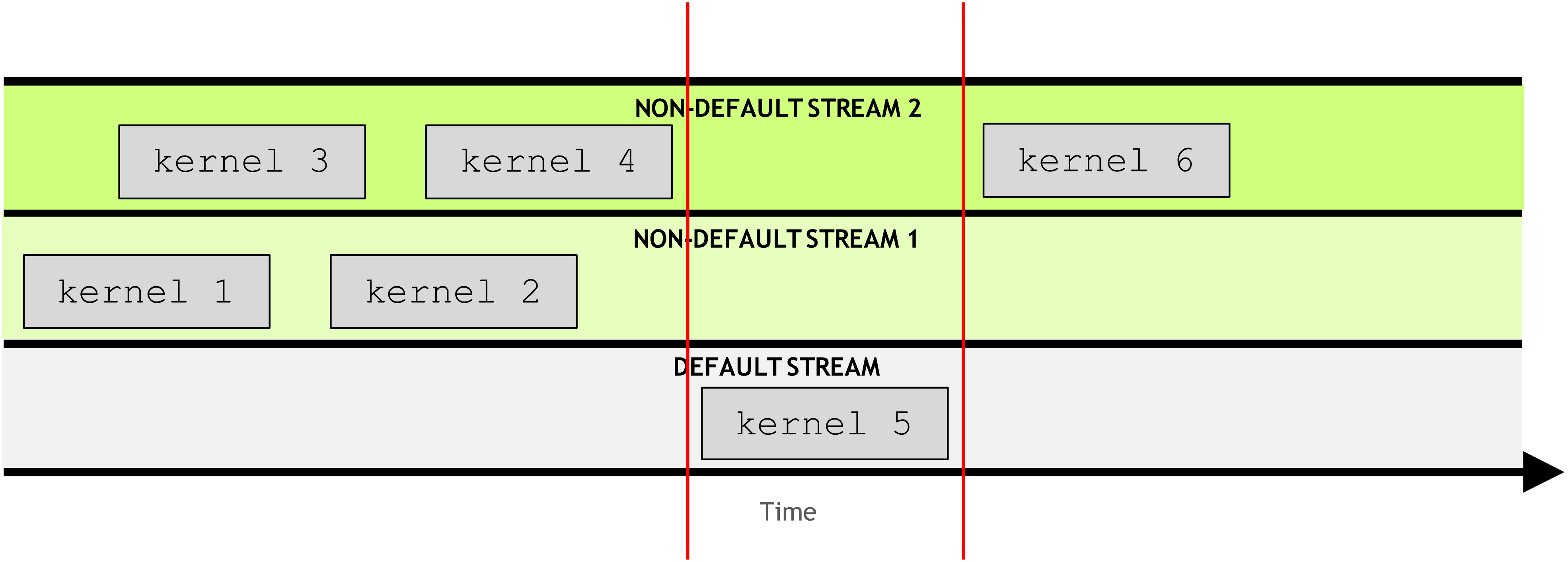
The default stream is special: **it blocks all kernels in all other streams**



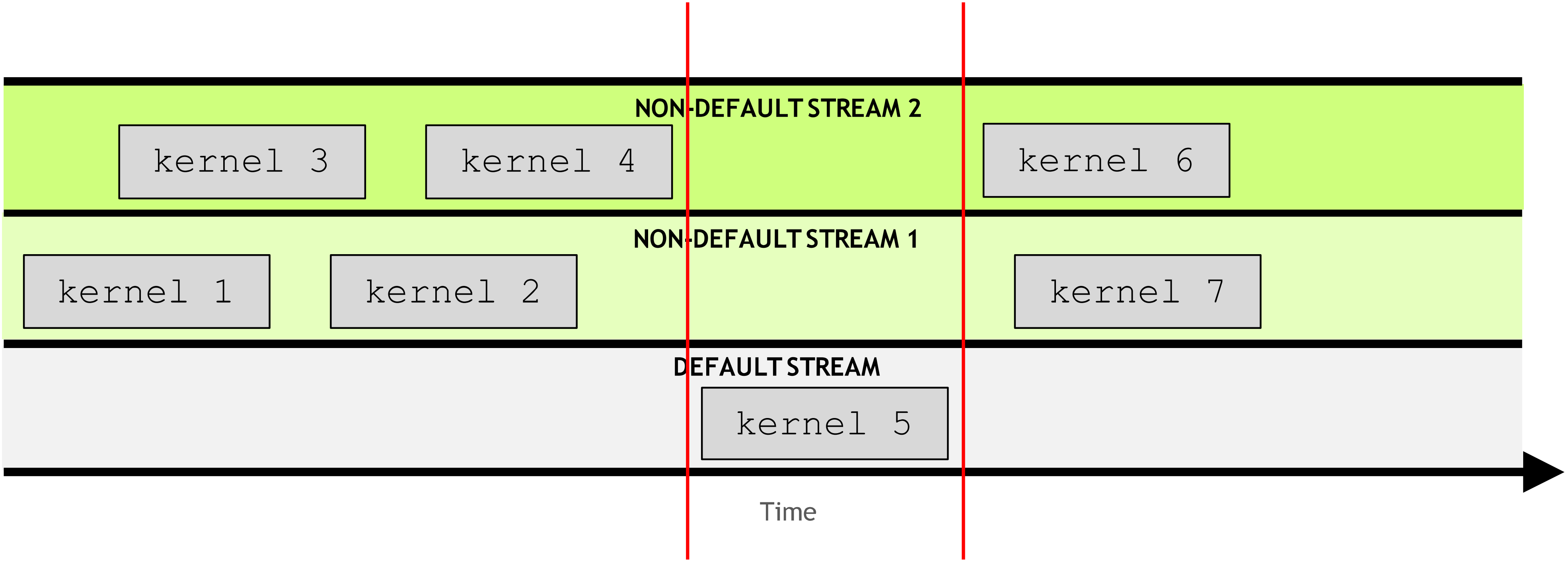
The default stream is special: **it blocks all kernels in all other streams**



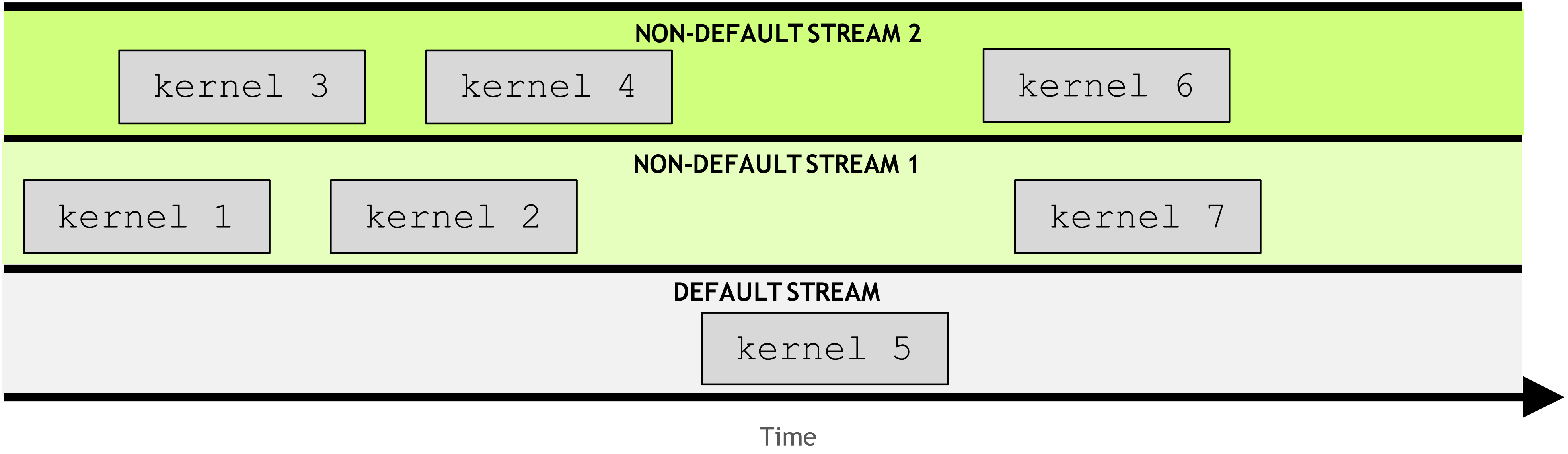
The default stream is special: **it blocks all kernels in all other streams**



The default stream is special: **it blocks all kernels in all other streams**

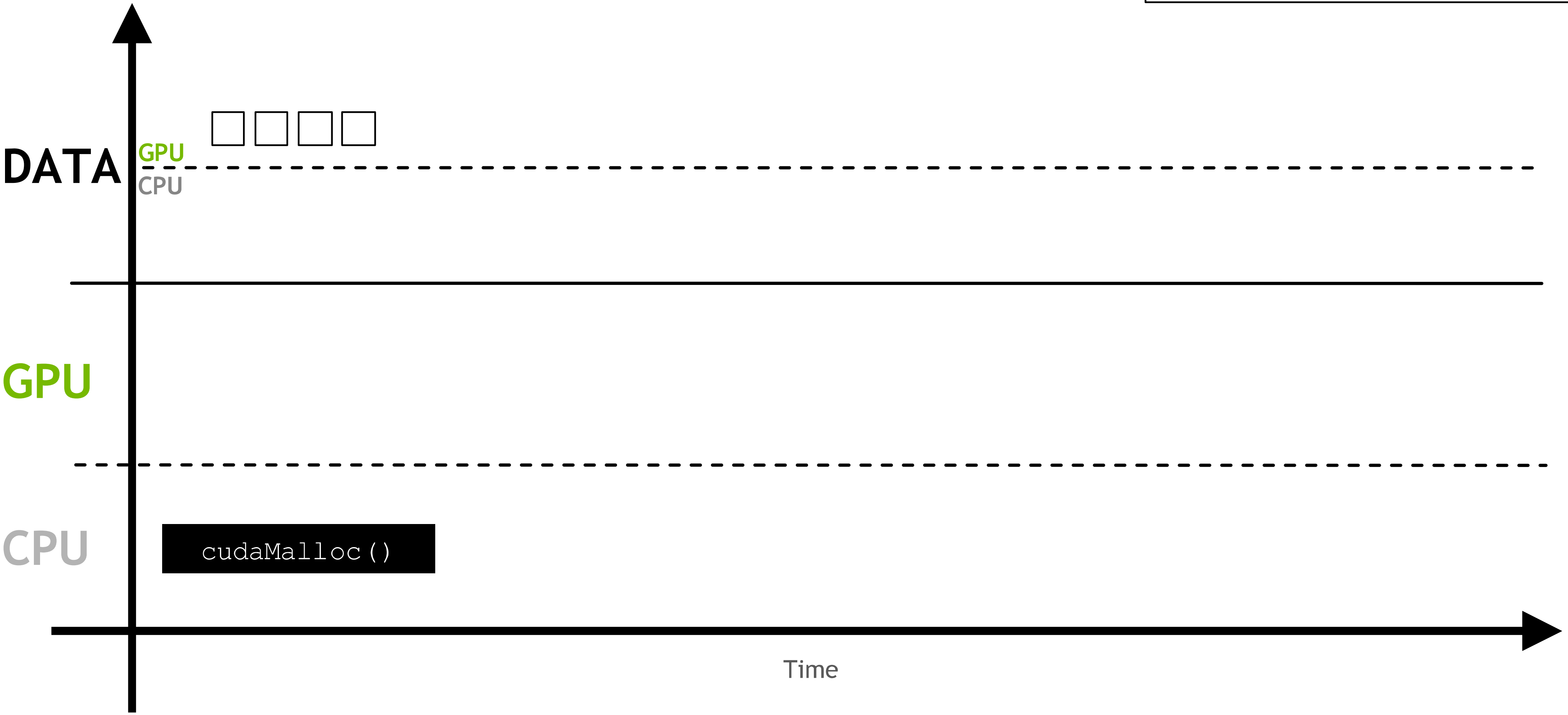


The default stream is special: **it blocks all kernels in all other streams**

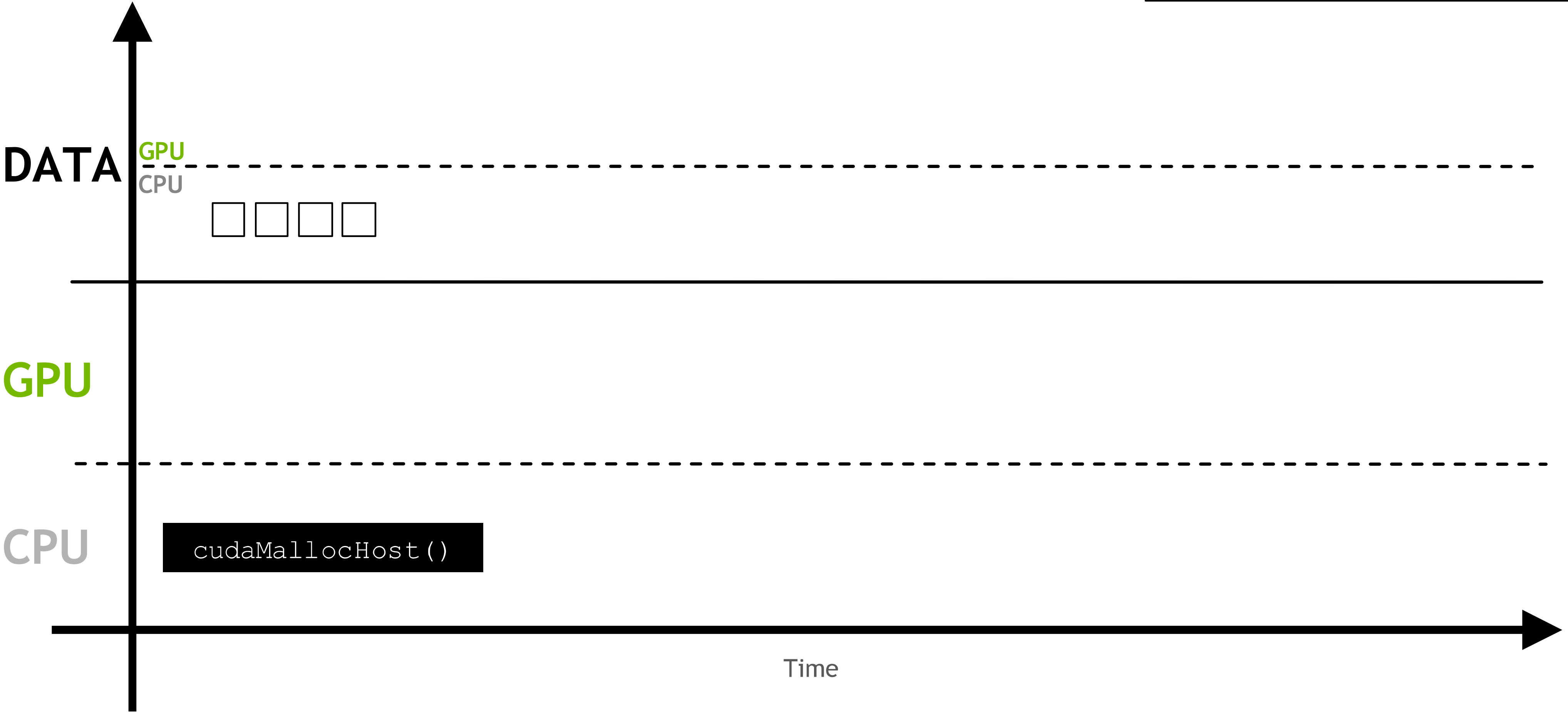


Non-Unified Memory

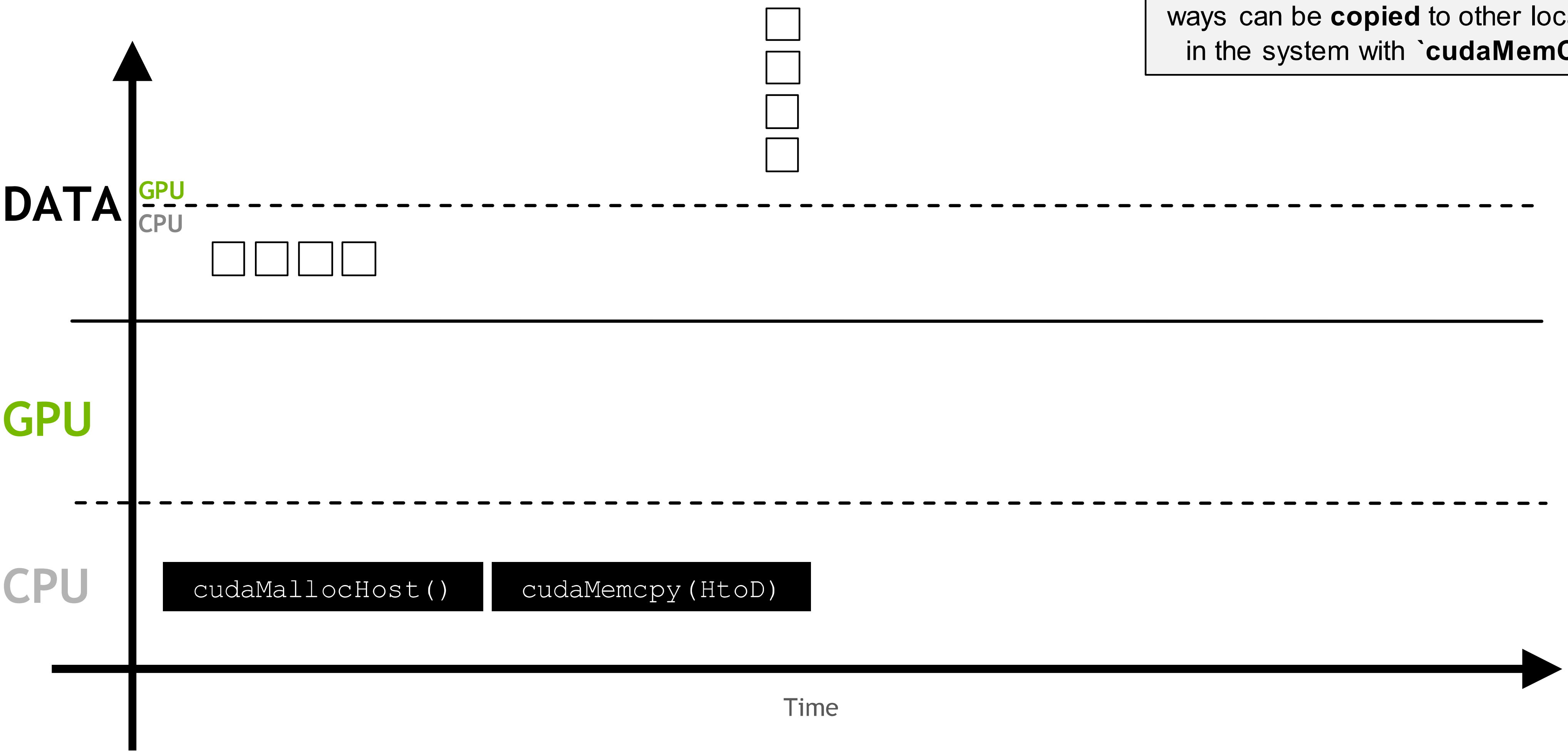
Memory can be allocated directly to the GPU with `**cudaMalloc**`



Memory can be allocated directly to the host with **`cudaMallocHost`**



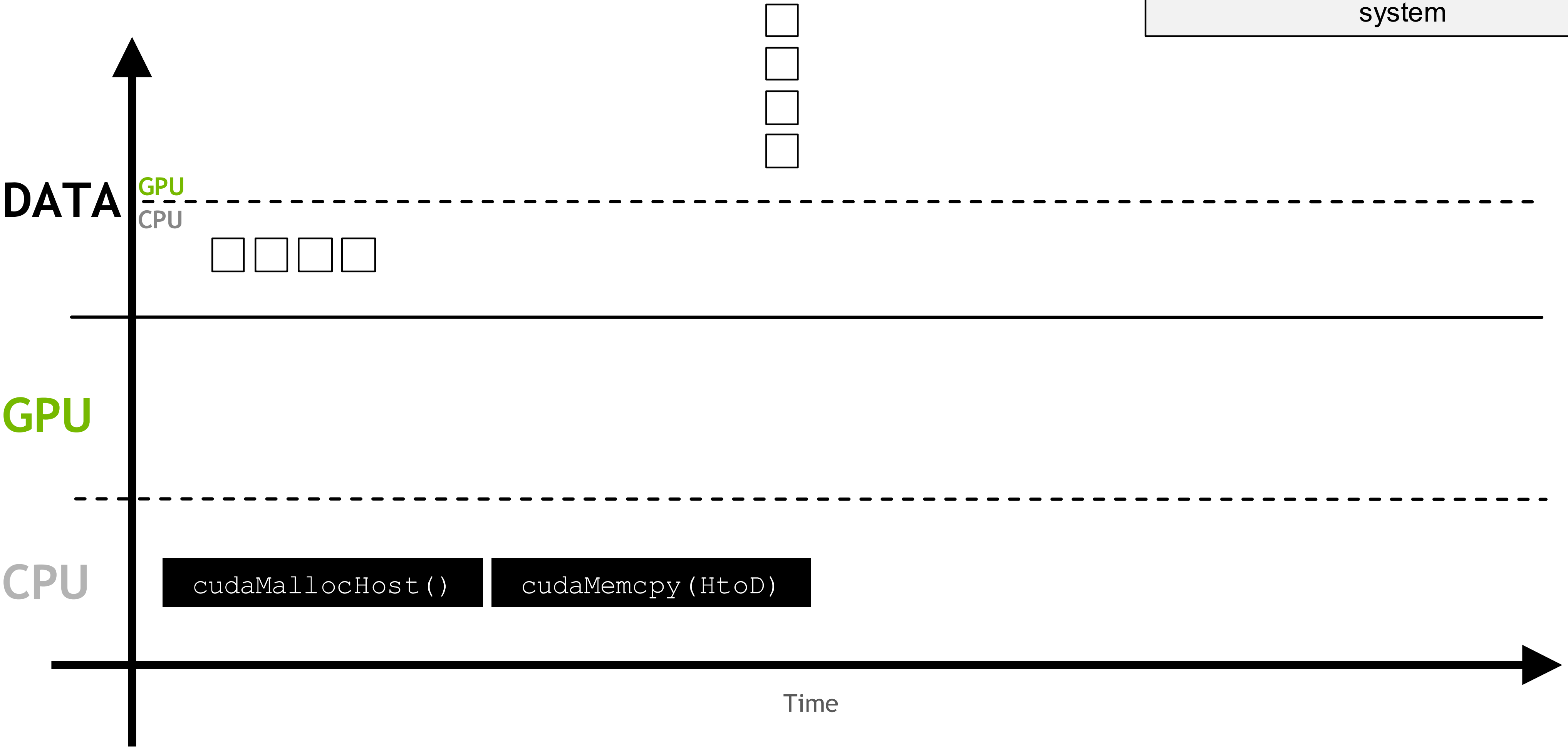
Memory allocated in either of these ways can be **copied** to other locations in the system with **`cudaMemCpy`**



cudaMallocHost()

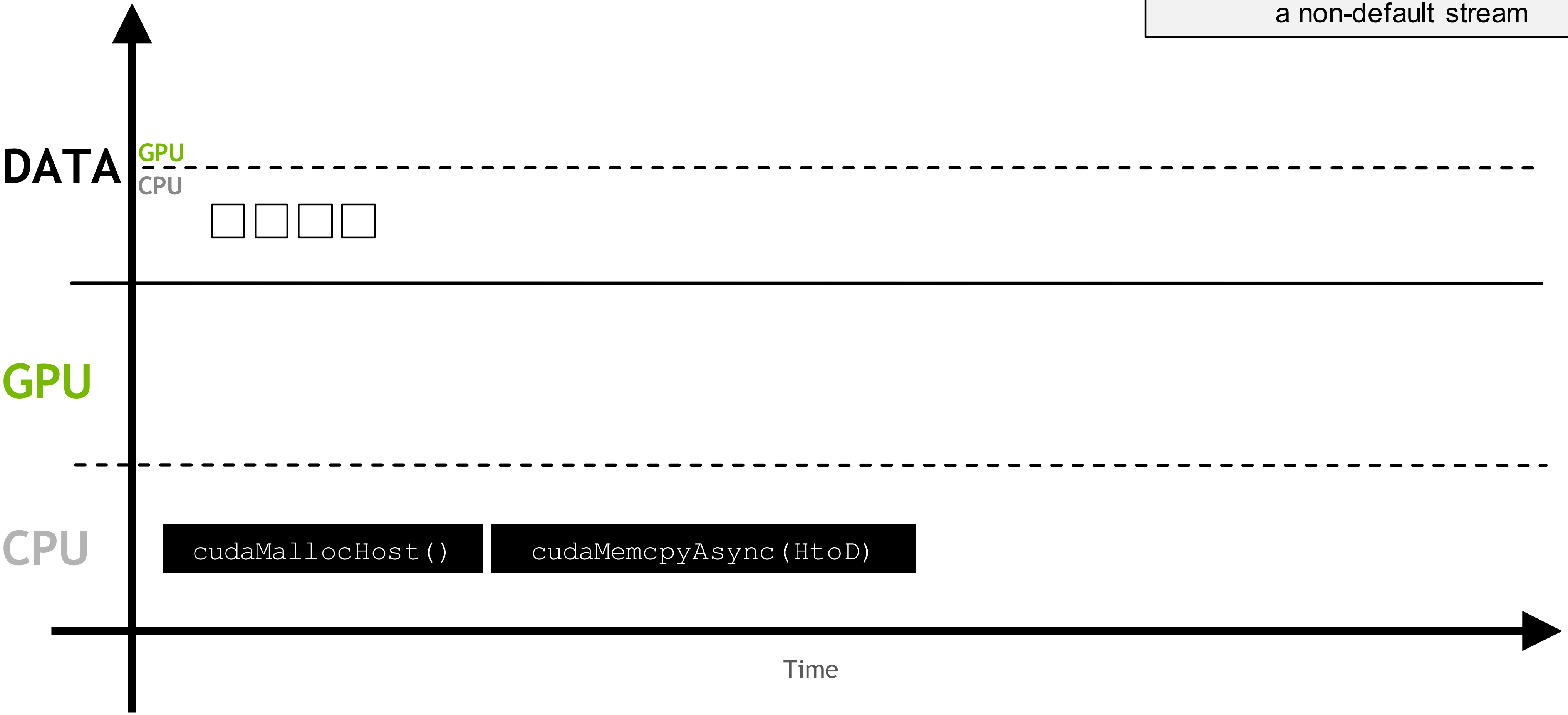
cudaMemcpy(HtoD)

Copying leaves 2 copies in of in the system

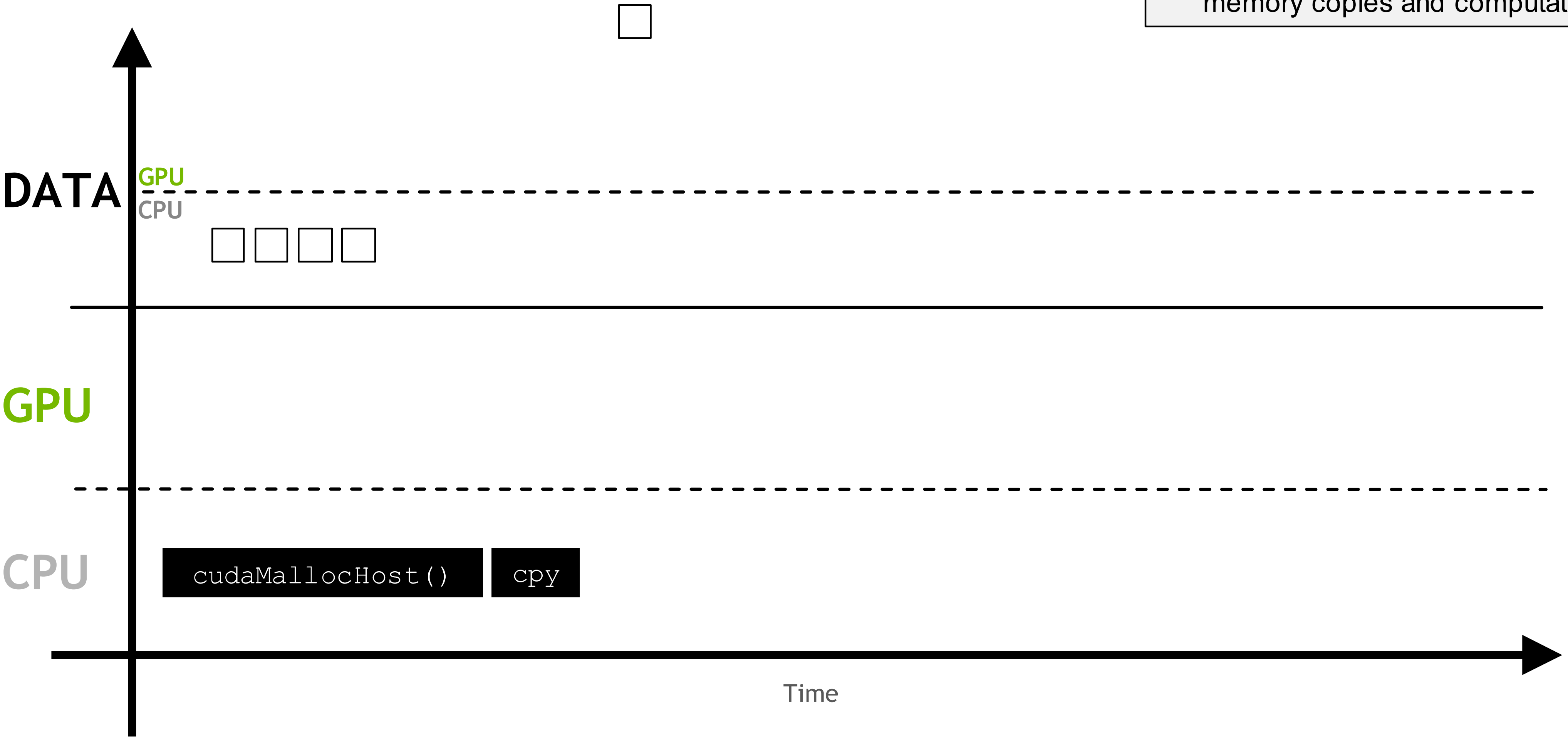


cudaMemcpyAsync

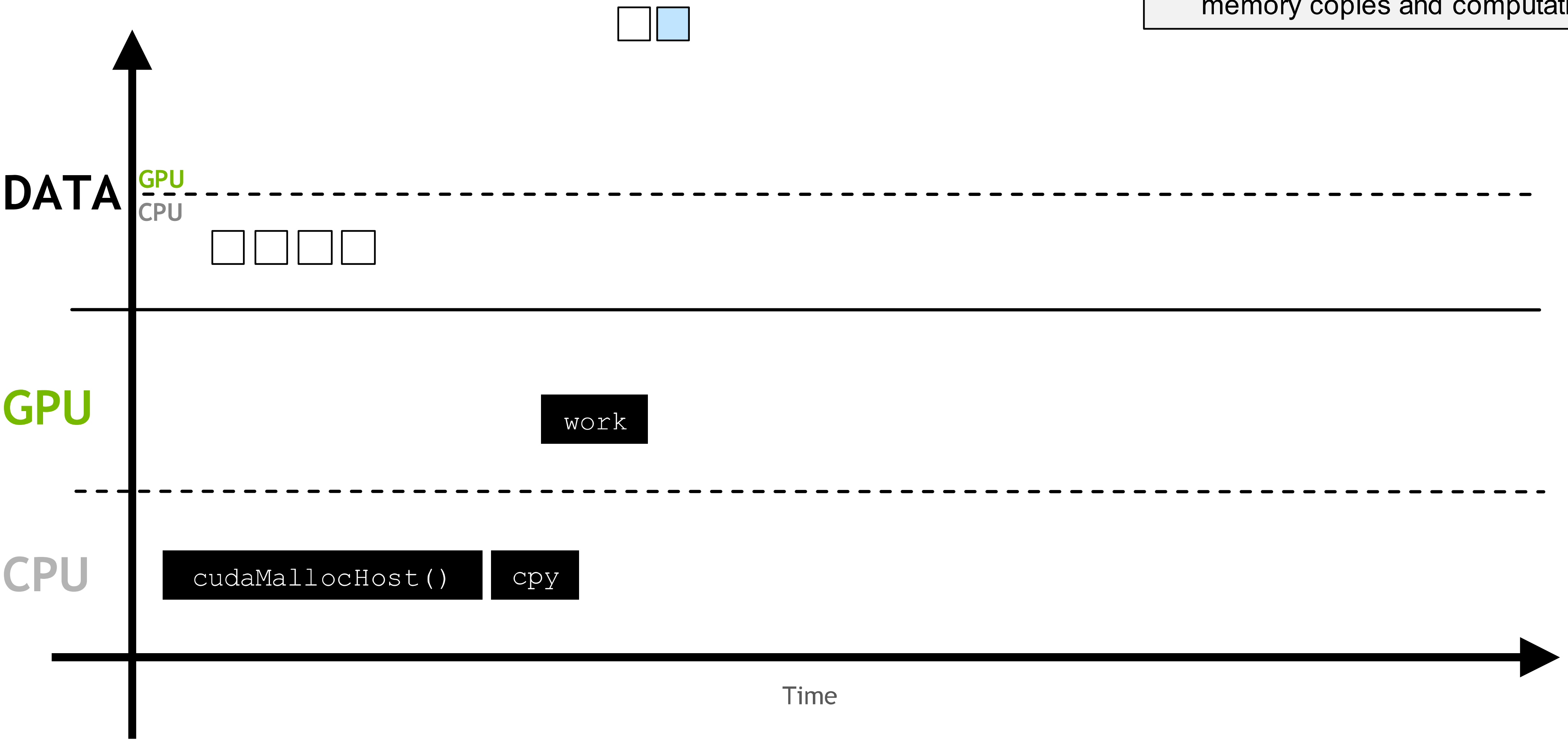
`cudaMemcpyAsync` can asynchronously transfer memory over a non-default stream



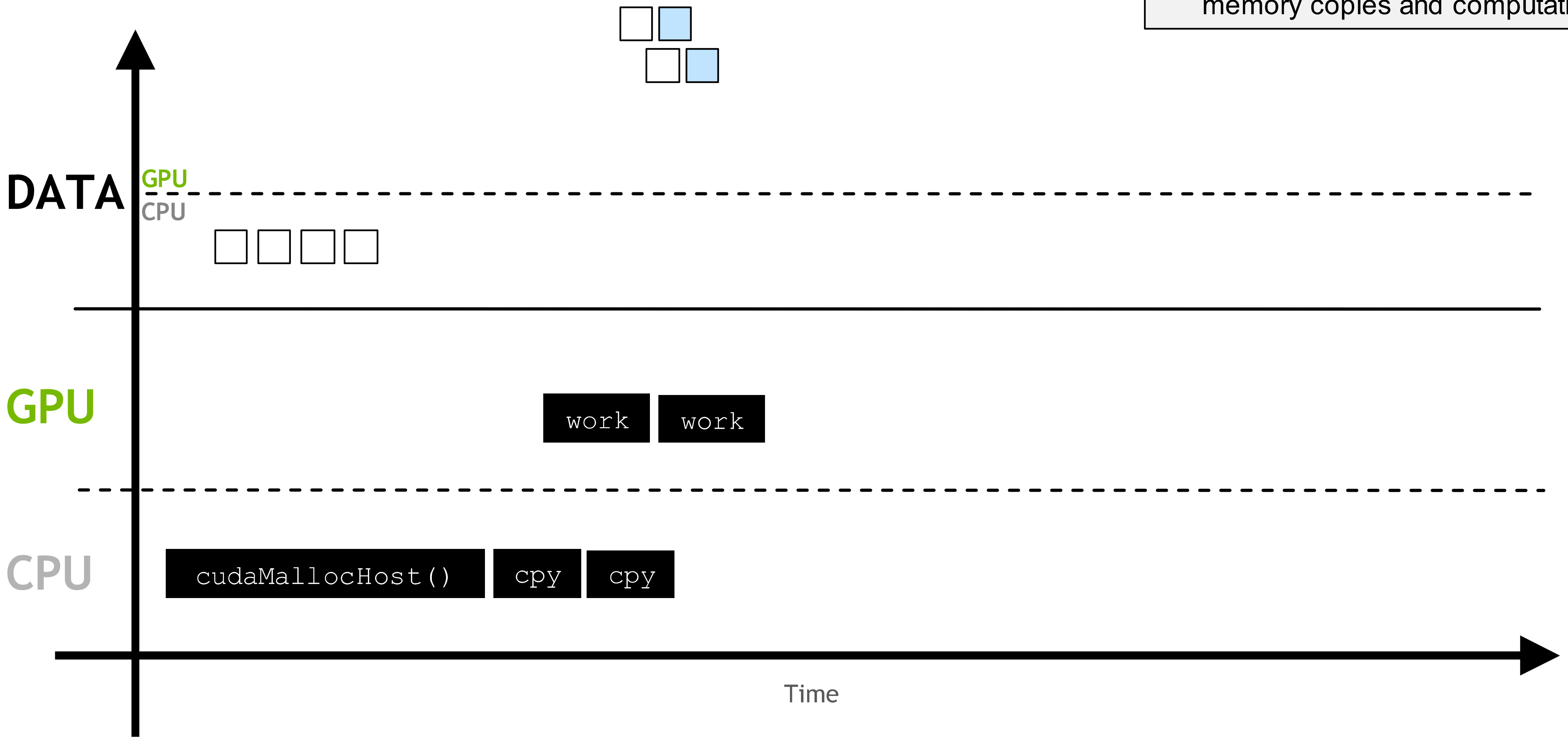
This can allow the **overlapping** memory copies and computation



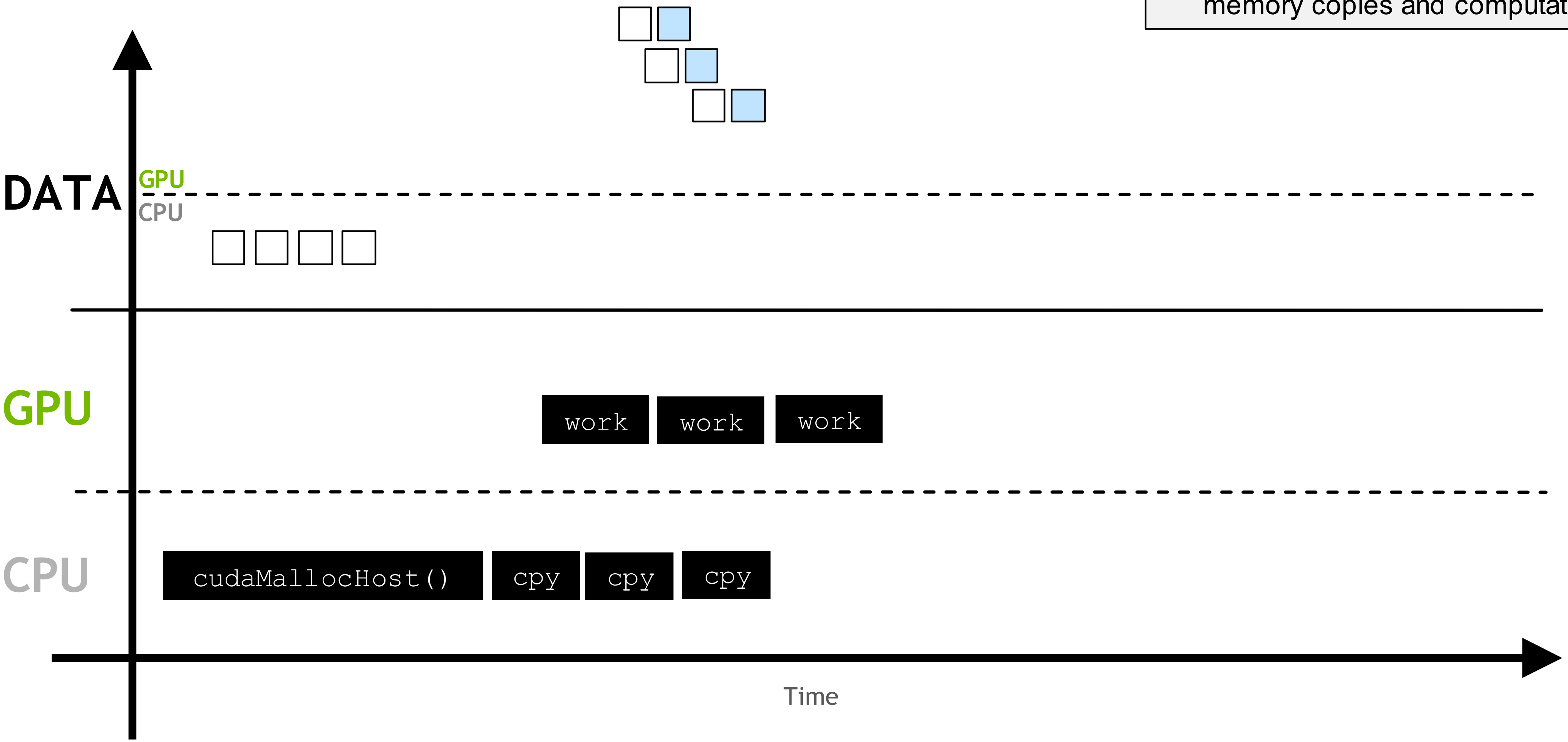
This can allow the **overlapping** memory copies and computation



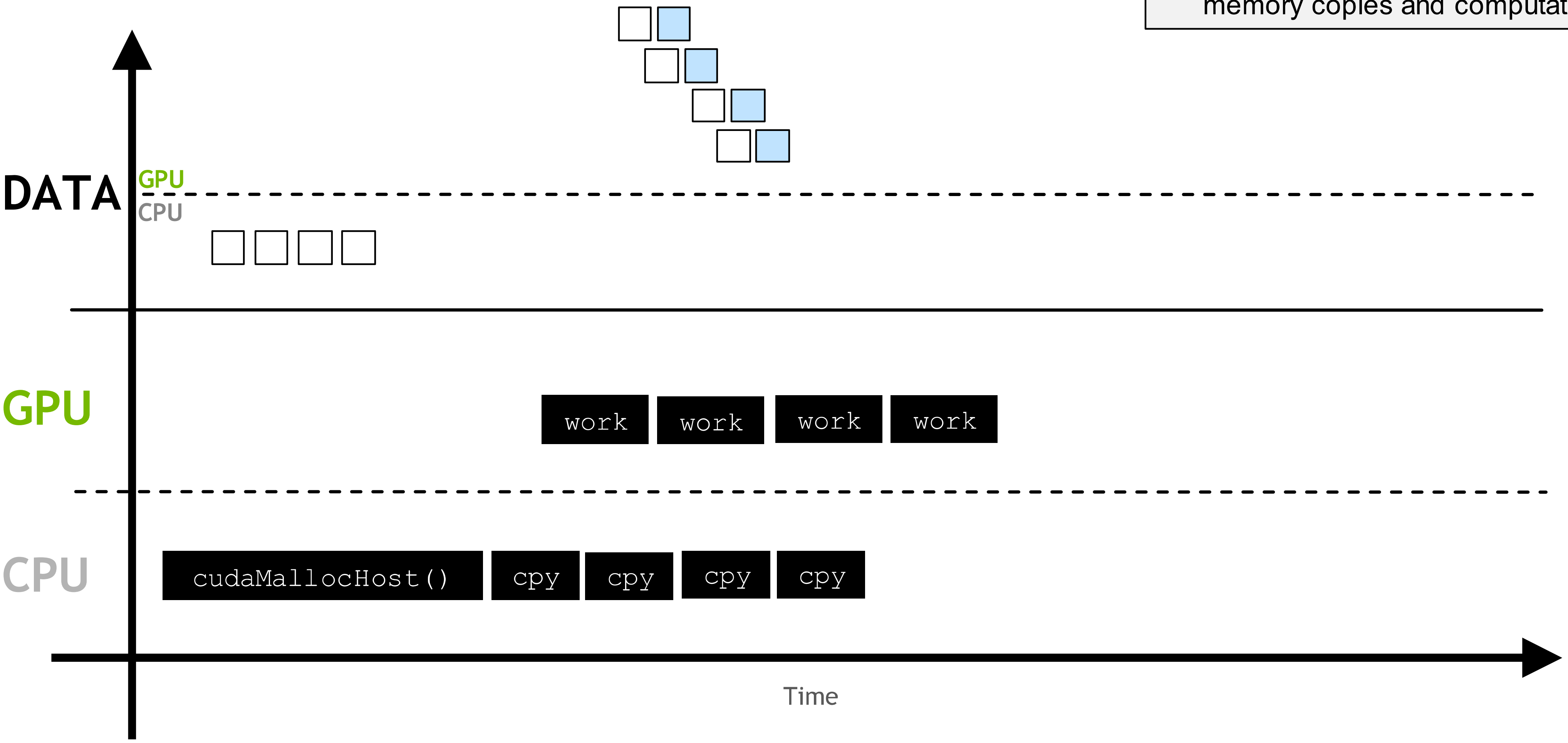
This can allow the **overlapping** memory copies and computation



This can allow the **overlapping** memory copies and computation



This can allow the **overlapping** memory copies and computation



NVIDIA

DEEP
LEARNING
INSTITUTE

www.nvidia.com/dli