Distributive

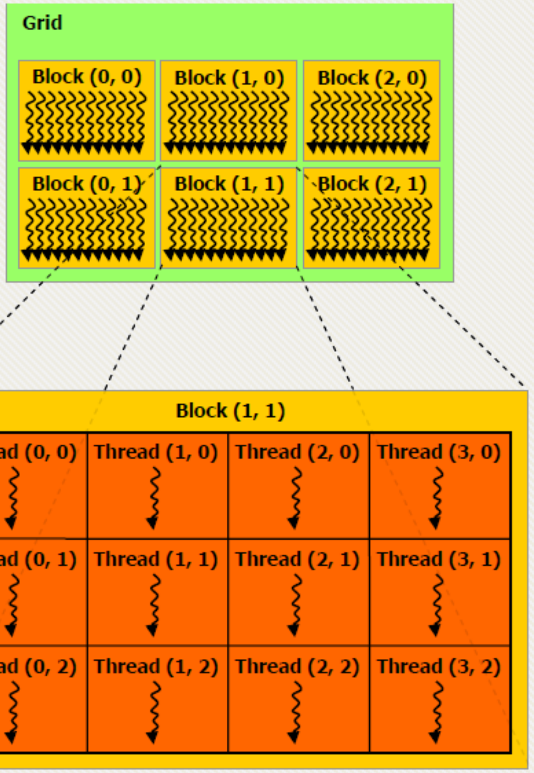
Introduction to GPUs - part 2
Streams and profiling

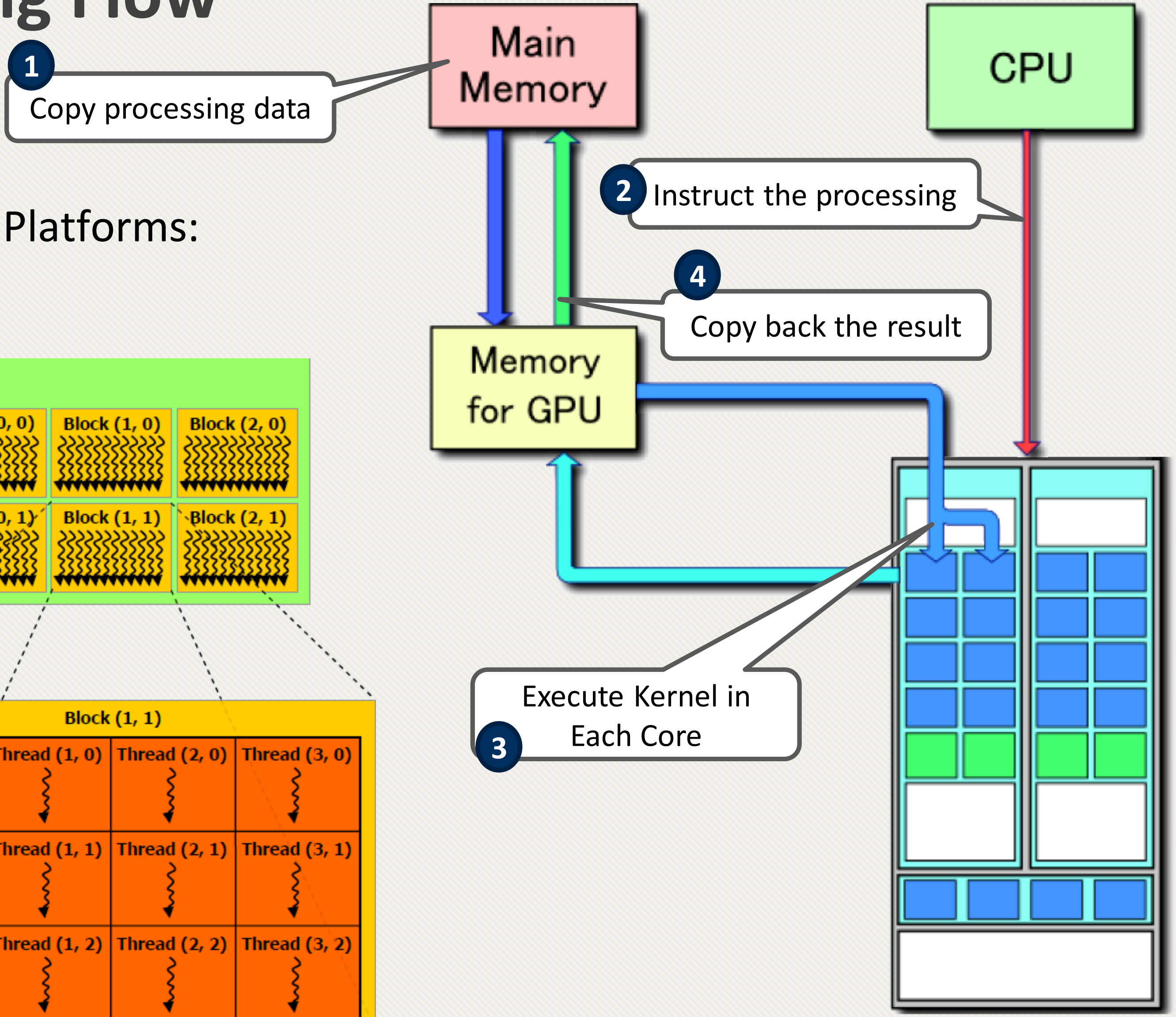
Distributive Book Club

October 5th , 2022

Reminder: GPU Processing Flow

- How can we run a code in GPU?
- There are very Mature environments & Platforms:
  - CUDA, OpenCL, …
- Keywords to remember:
  - SM (Streaming Multiprocessor)
  - ThreadBlock & GridBlock
  - Warp (SIMD execution)
  - Kernel





CUDA Features

1. Shared Memory
2. **CUDA Streams -> Today's talk**
3. Unified Memory
4. Dynamic Parallelism
5. Hyper-Q
6. Warp-Level Primitives (Shuffle Instructions)
7. MPS
8. NCCL library (It's not a feature!)
9. Cooperative Groups
10. CUDA Graphs
11. Multi Instance GPU (MIG)
12. Async-Copy
13. Thread Collectives
14. C++17 STL, templates, pointer aliasing…

Concurrency Using Streams

- The ability to perform multiple CUDA operations simultaneously (beyond multi-threaded parallelism)
  - cudaMemcpyAsync (HostToDevice)
  - CUDA Kernel <<<>>>
  - cudaMemcpyAsync (DeviceToHost)

- Stream:
  - A sequence of operations that execute in issue-order on the GPU

- Programming model used to effect concurrency
  - CUDA operations in different streams may run concurrently
  - CUDA operations from different streams may be interleaved

Concurrency Using Streams

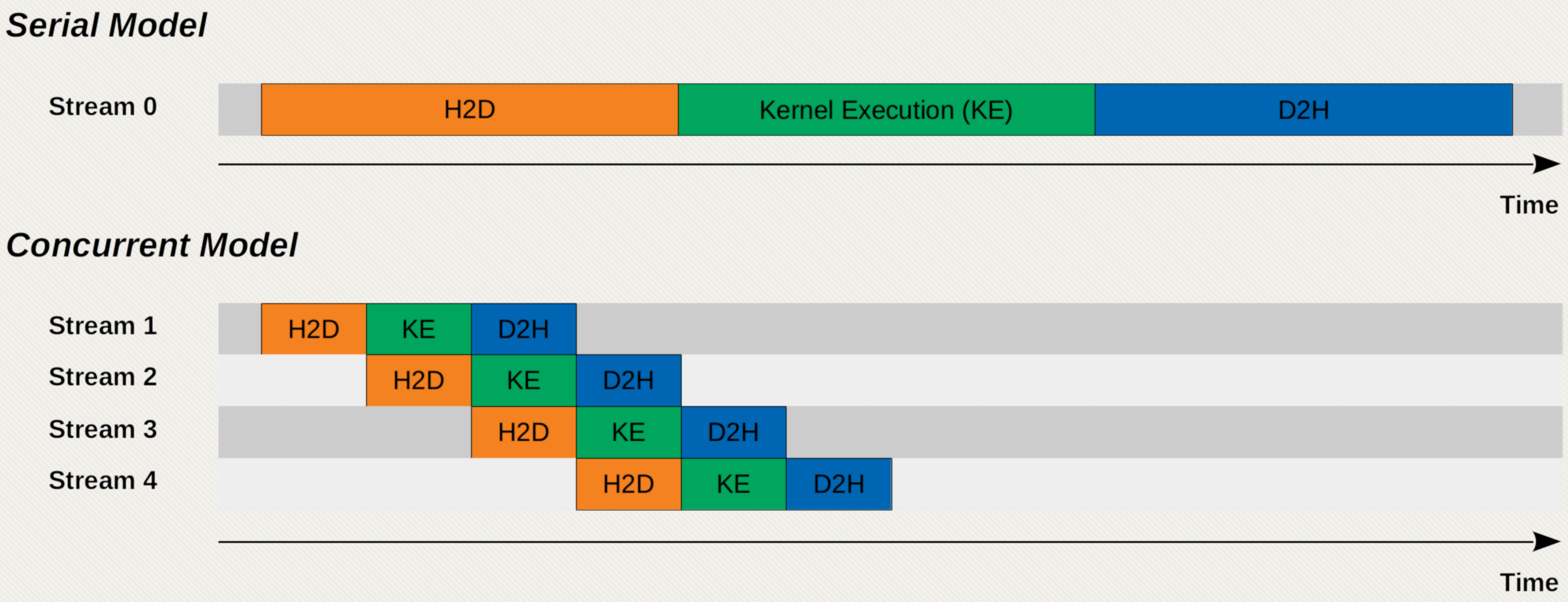


Image source: https://leimao.github.io/blog/CUDA-Stream/

Let’s see some demo and speedup!

- Single stream vector operation vs. Multiple-stream vector operation

https://github.com/amirsojoodi/GPGPU-Course/tree/master/13th/CUDA/streams

Let’s Profile/Monitor the GPU(s)

- Nsight systems
- Multiple ways of data gathering
  - Launch the app from the profiler:

```
$ nsys profile --trace=cuda -o output-report ./path/to/file
```

  - We can also launch the profiler separately which listens for other processes.
  - Then feed the report to the Nsight Systems GUI
- Other commands:
  - Monitoring the GPUs from the command line

```
$ nvidia-smi pmon
```

https://docs.nvidia.com/nsight-systems/UserGuide/index.html
https://github.com/amirsojoodi/Manuals-and-Tutorials/tree/master/Programming/CUDA

NVIDIA Hyper-Q

- Multiple work queues between the host and the GPU
- Hardware support for stream concurrency even at the issuing-time!

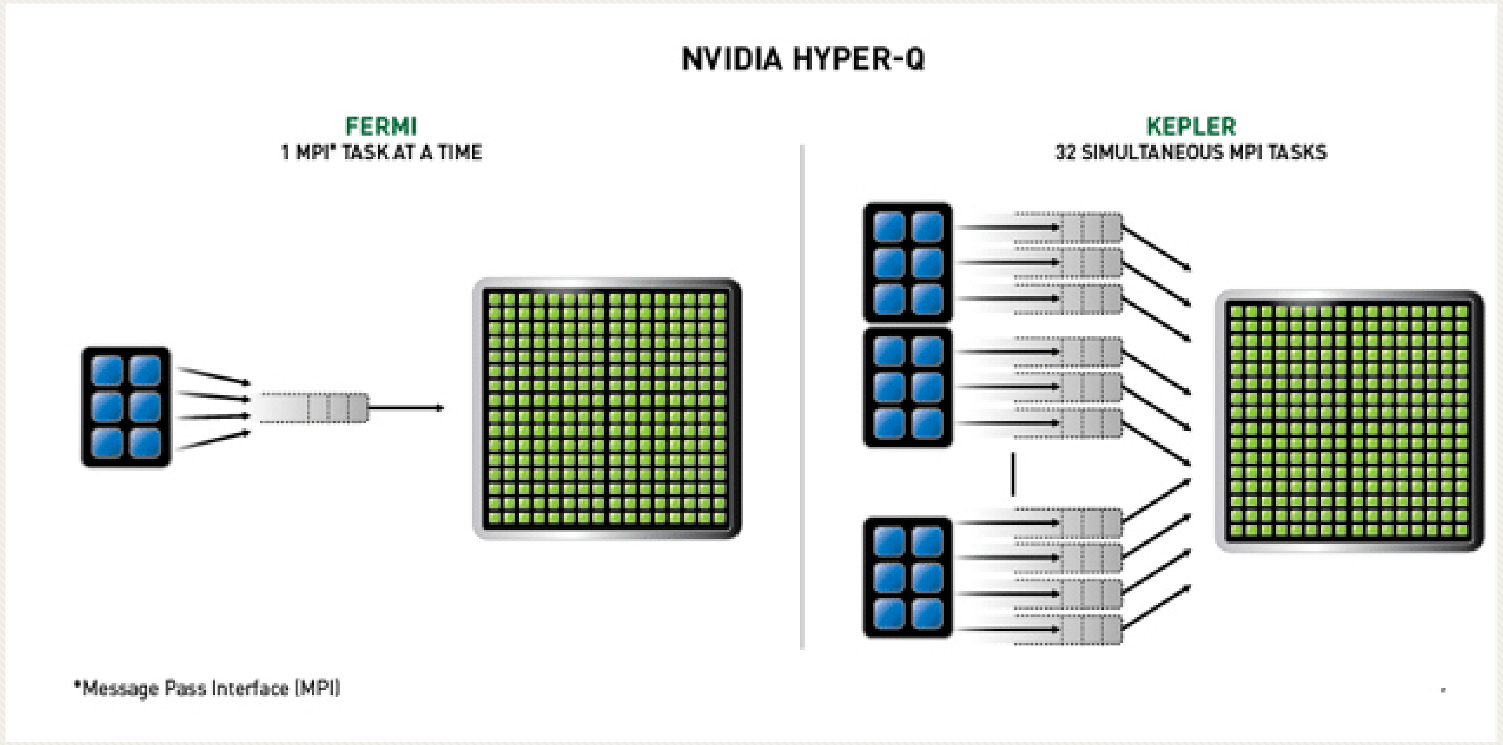


Image source: NVIDIA

Thank You ☺

Instead of blaming darkness, let’s light a candle!

Questions, Comments, and Ideas are Welcome!