Parallel Computing with GPUs

Introduction to CUDA
Part 1 – Programming Model

The University Of Sheffield.

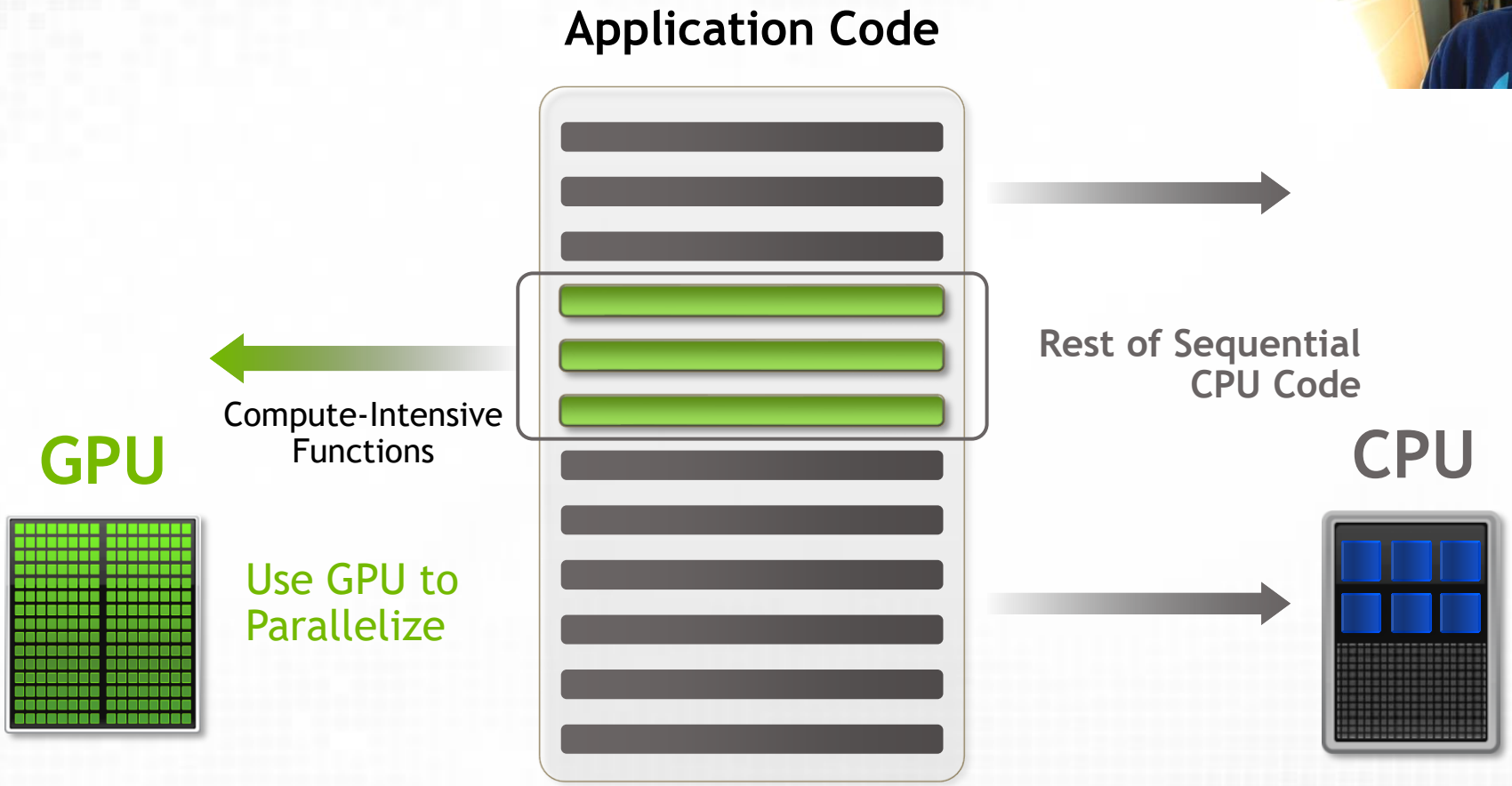
Dr Paul Richmond

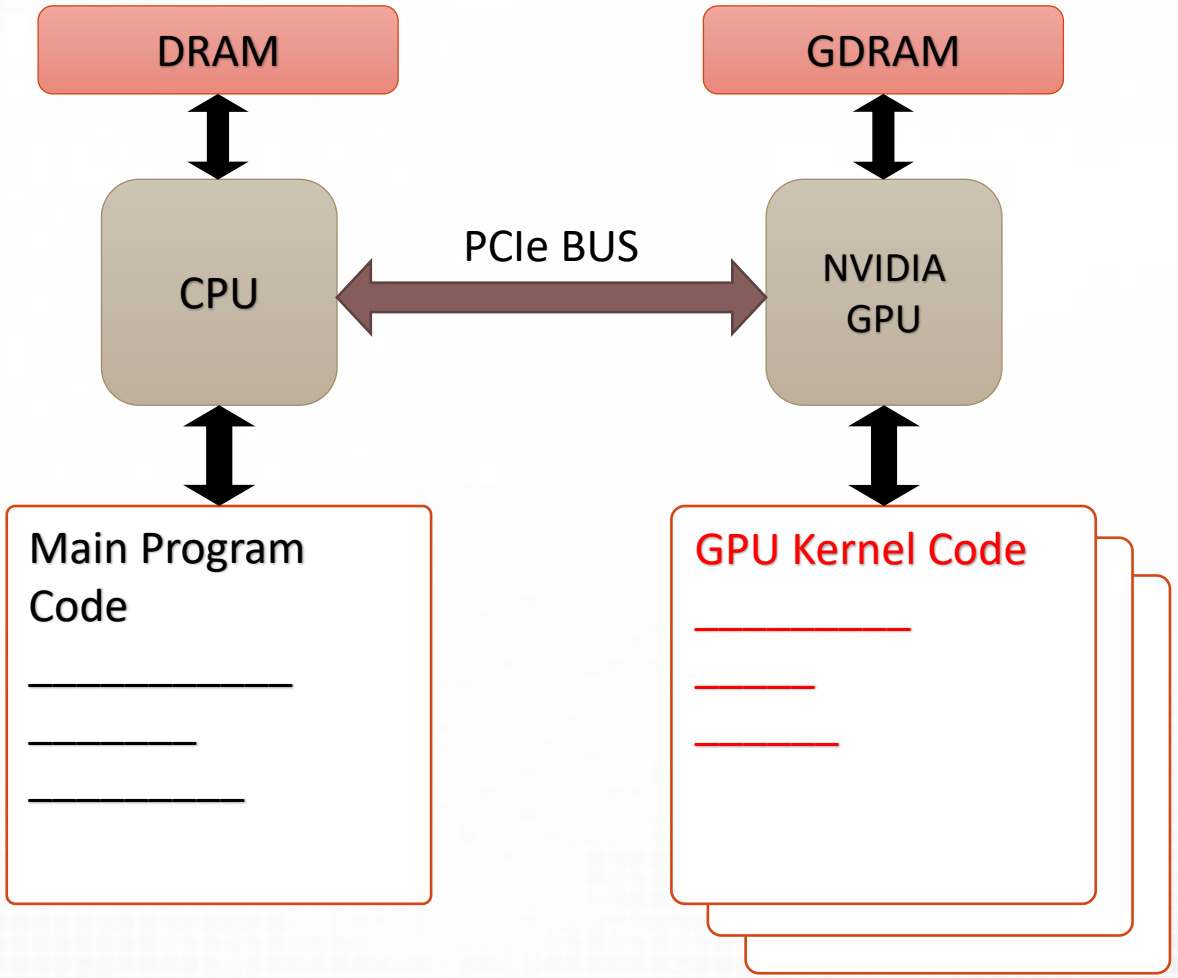
http://paulrichmond.shef.ac.uk/teaching/COM4521/

This Lecture (learning objectives)

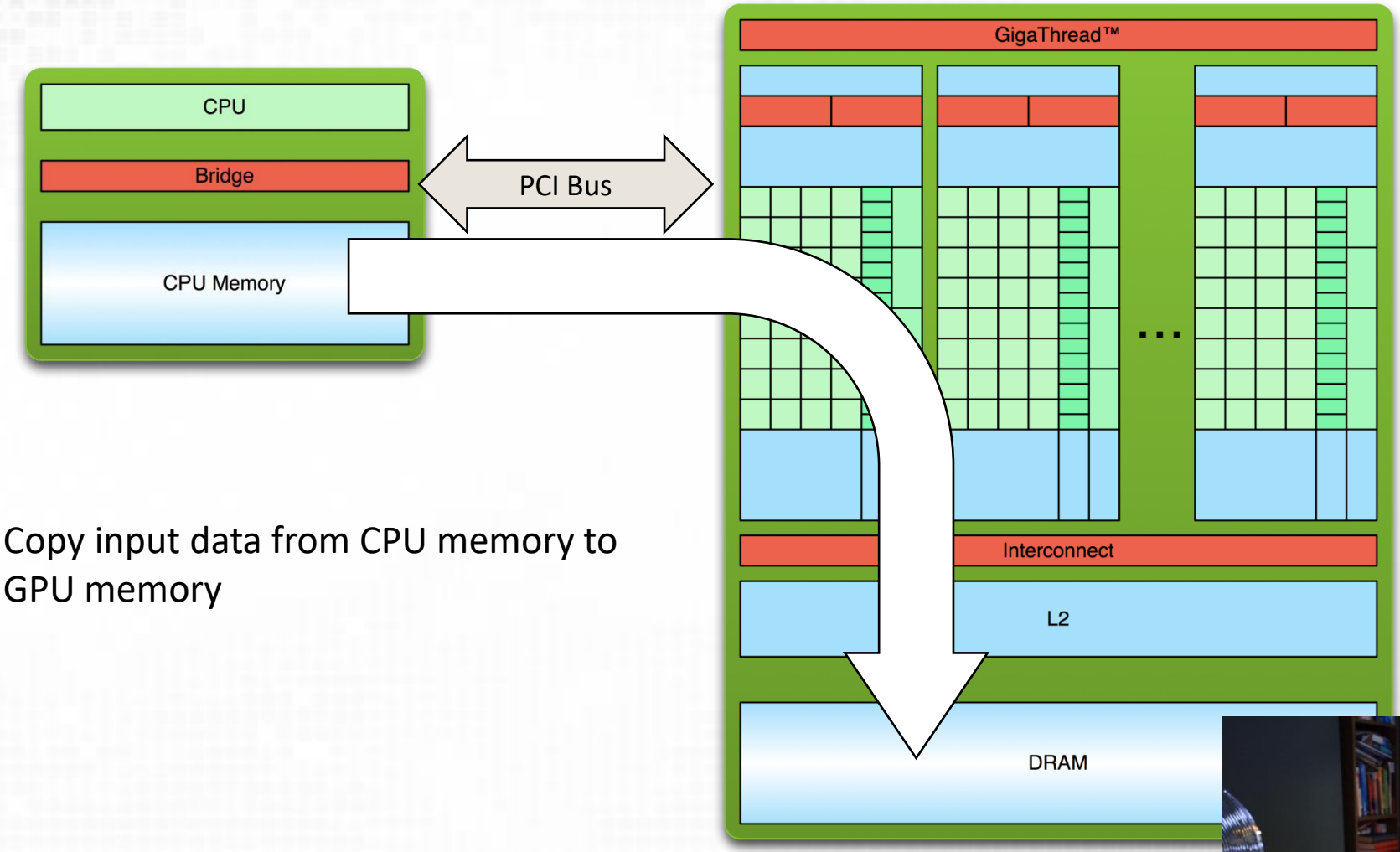
- CUDA Programming Model
  - Present the processing flow for running a GPU program
  - Explain the CUDA software model and its relation to the hardware hierarchy
  - Propose a simple problem which can be implemented on the GPU

Programming a GPU with CUDA

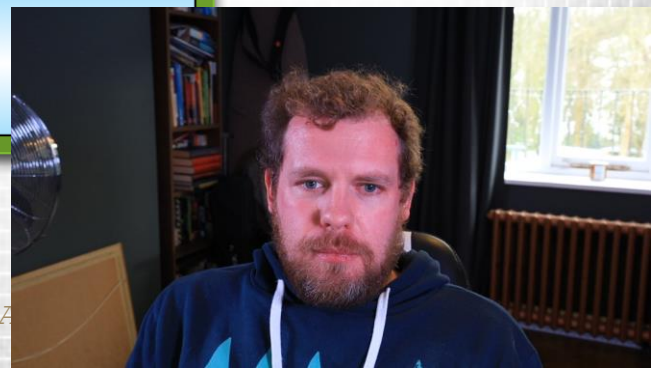



DRAM
GDRAM
CPU
PCIe BUS
NVIDIA
GPU
Main Program
Code
GPU Kernel Code

Simple processing flow

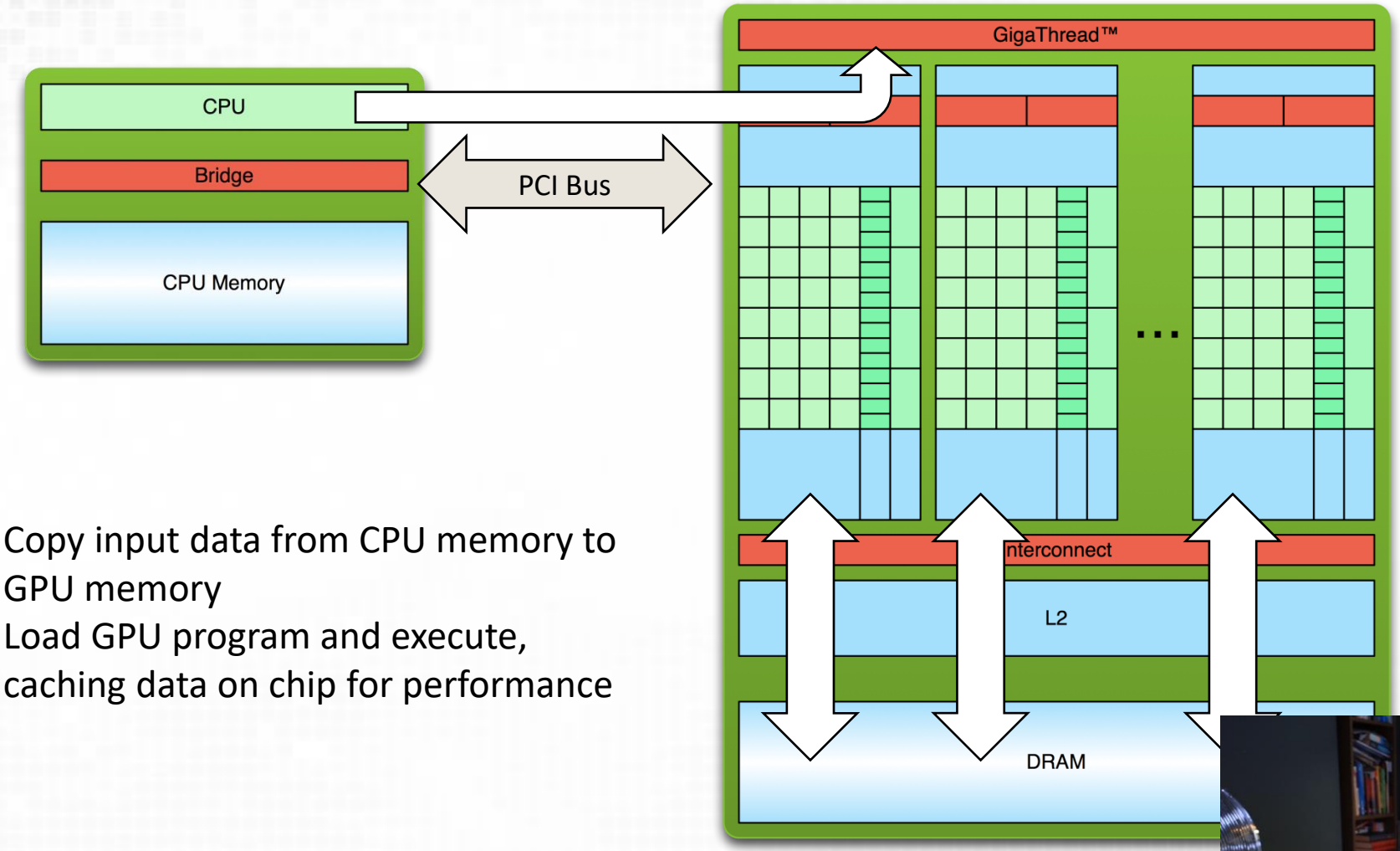


1. Copy input data from CPU memory to GPU memory

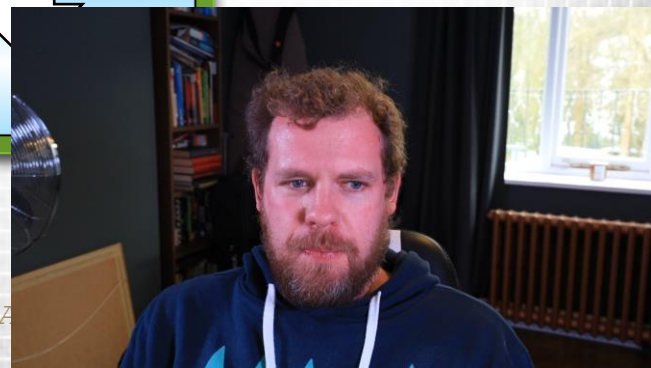


Images: Mark Harris (NVIDIA

Simple processing flow

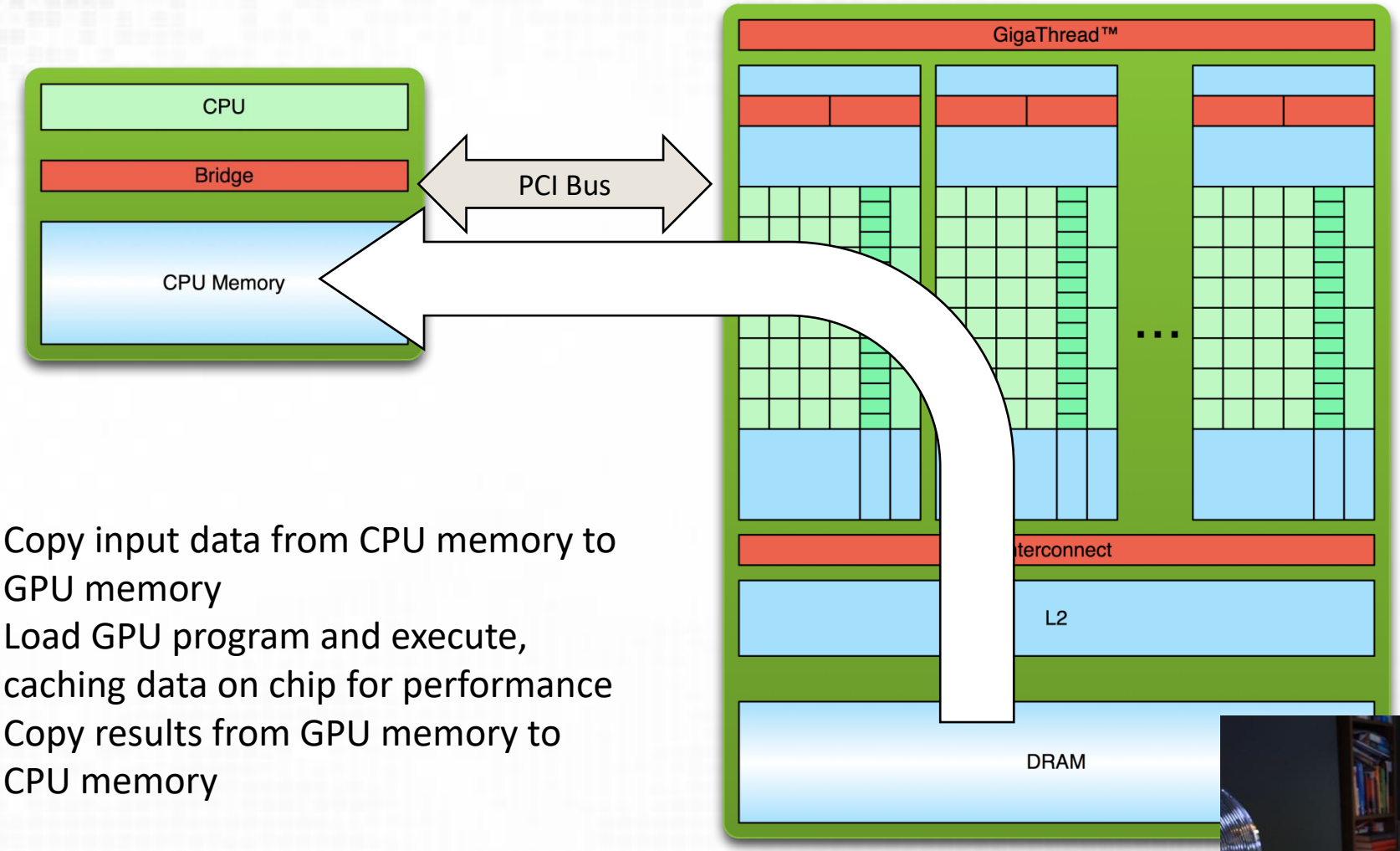


1. Copy input data from CPU memory to GPU memory
2. Load GPU program and execute, caching data on chip for performance

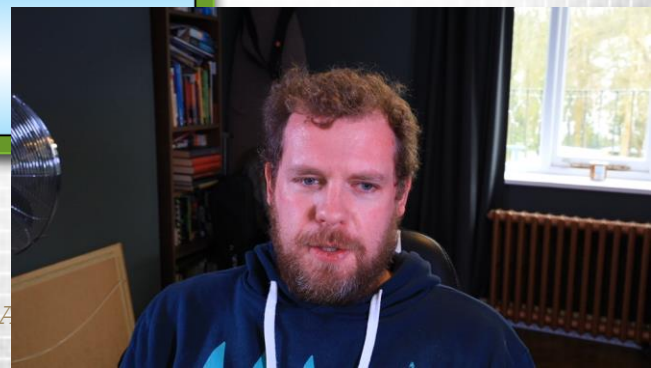


Images: Mark Harris (NVIDIA

Simple processing flow

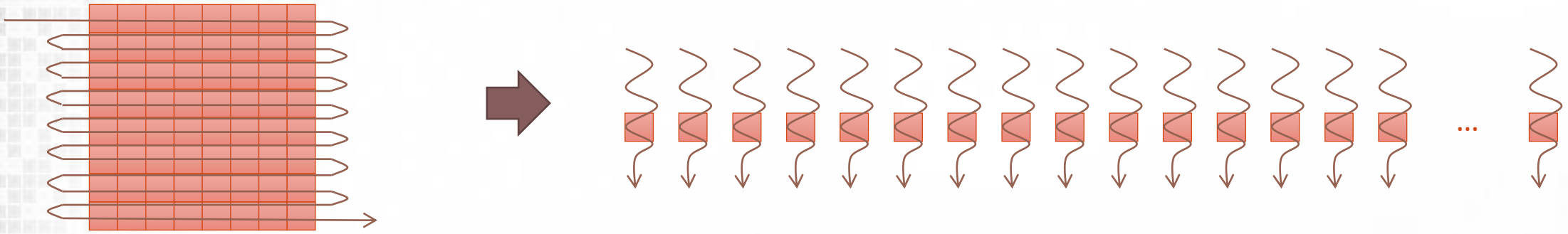


1. Copy input data from CPU memory to GPU memory
2. Load GPU program and execute, caching data on chip for performance
3. Copy results from GPU memory to CPU memory



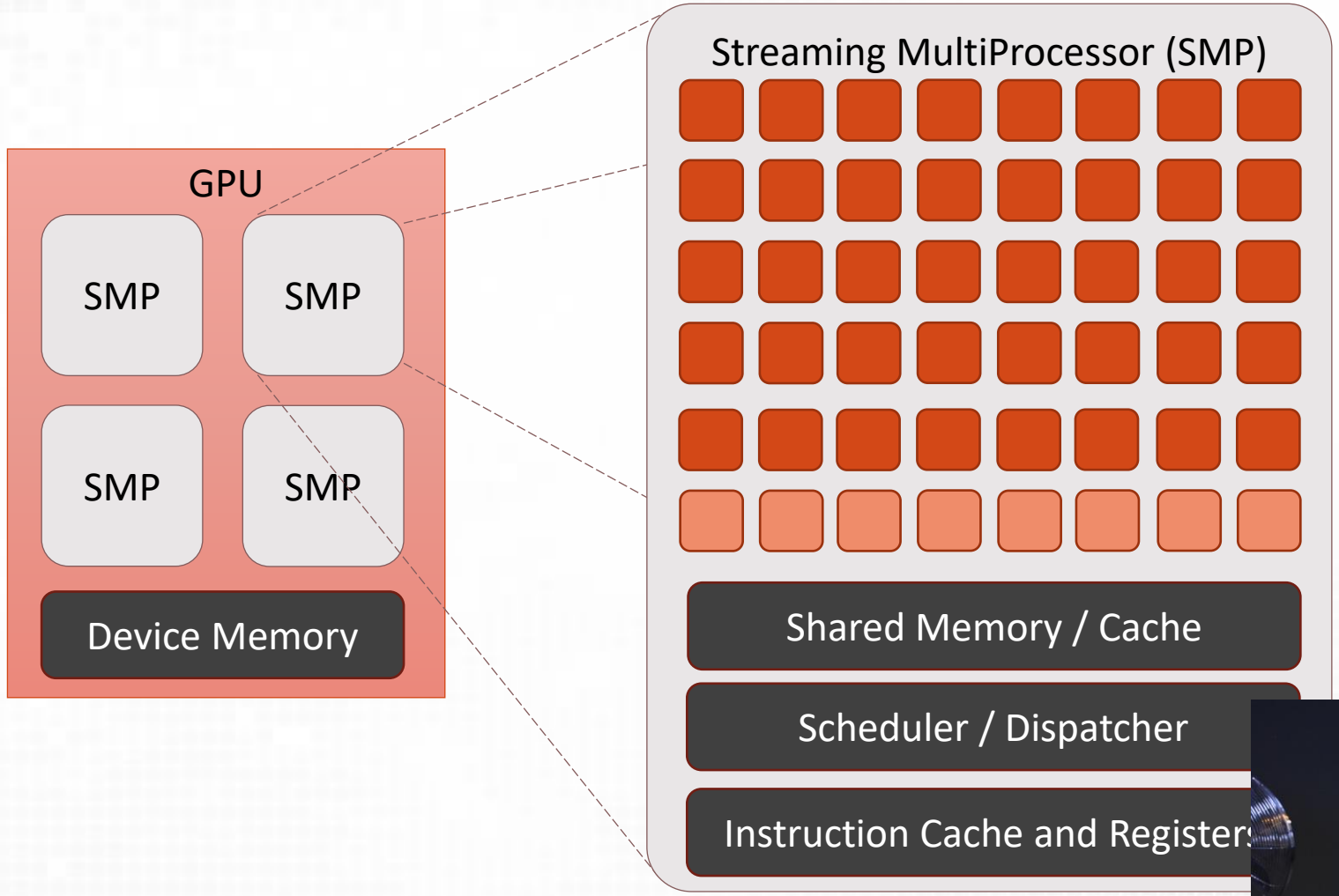
Images: Mark Harris (NVIDIA

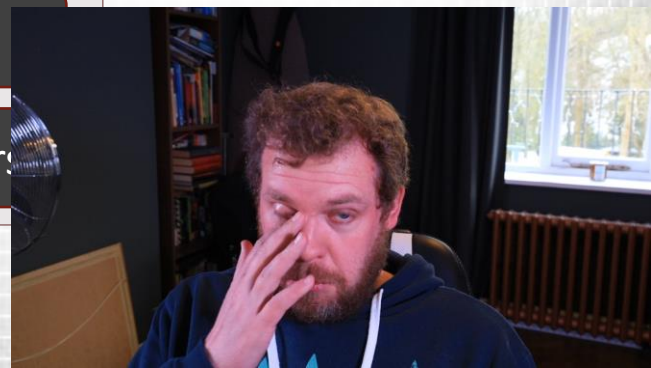
Stream Computing



- Data set decomposed into a **stream** of elements
- A single computational function (**kernel**) operates on each element
  - A **thread** is the execution of a kernel on one data element
- Multiple Streaming Multiprocessor cores can operate on multiple elements in parallel
  - Many parallel threads
- Suitable for **Data Parallel** problems

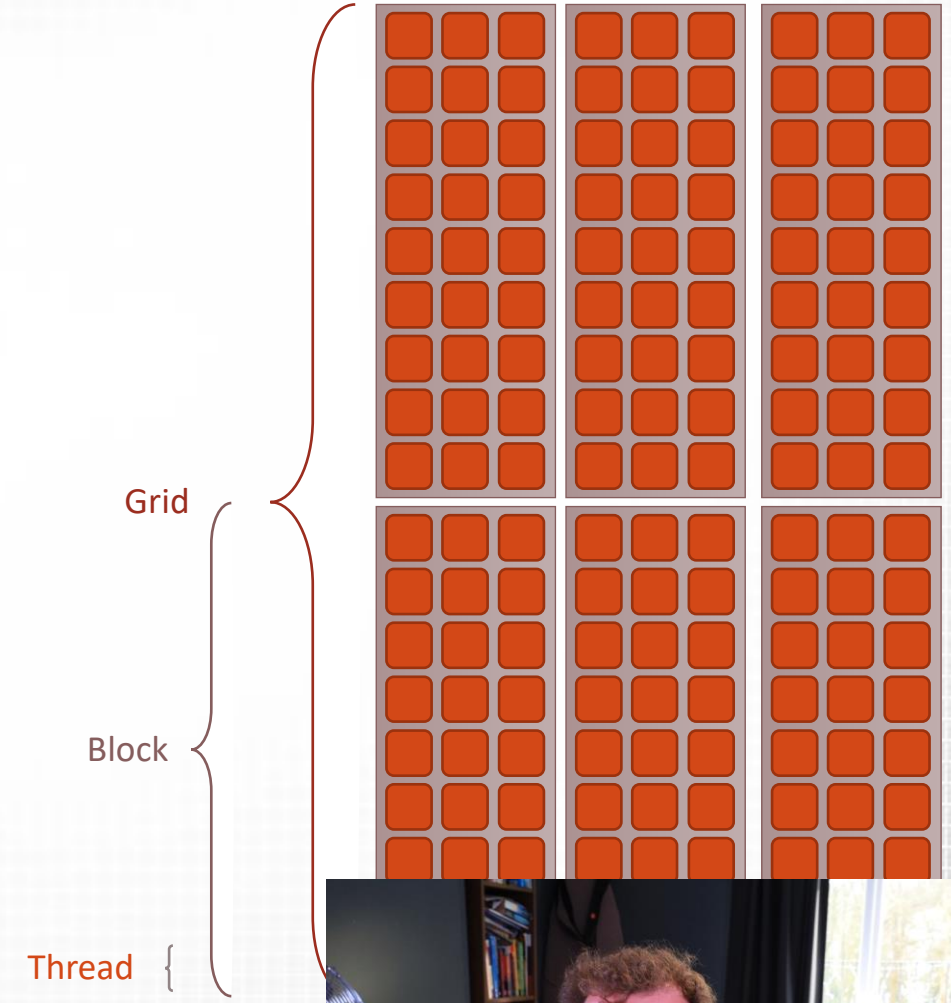
- How does the stream computing principle map to the with the hardware model?





CUDA Software Model

- Hardware abstracted as a **Grid** of **Thread Blocks**
  - Blocks map to SMPs
  - Each thread maps onto a CUDA core
  - Blocks may be 1D, 2D or 3D
- Don’t need to know the hardware characteristics
  - Oversubscribe the device
  - Code is portable across different GPU versions



CUDA Vector Types

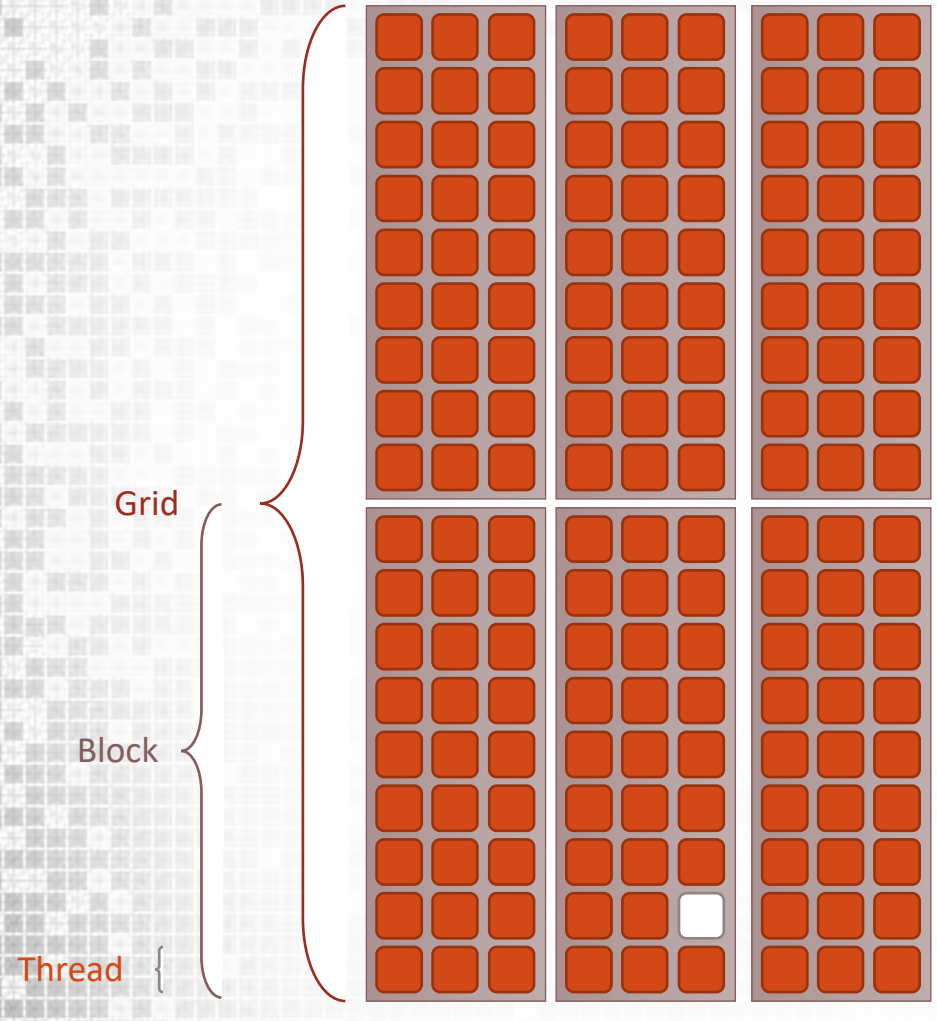
- CUDA Introduces a new `dim` types. E.g. `dim2`, `dim3`, `dim4`
  - `dim3` contains a collection of three integers (X, Y, Z)

```
dim3 my_xyz (x_value, y_value, z_value);
```

- Values are accessed as members

```
int x = my_xyz.x;
```

Special dim3 Vectors



- `threadIdx`
  - The location of a thread within a block. E.g. (2,1,0)
- `blockIdx`
  - The location of a block within a grid. E.g. (1,0,0)
- `blockDim`
  - The dimensions of the blocks. E.g. (3,9,1)
- `gridDim`
  - The dimensions of the grid. E.g. (3,2,1)

`Idx` values use zero indices, Dim values are a size


The University of Sheffield
CUDA , HALL OF RESIDENCE

Analogy

- Students arrive at halls of residence to check in
  - Rooms are already assigned in order
- Unfortunately admission rates are down!
  - Only half as many rooms as students
  - Each student can be moved from room i to room $2i$ so that no-one has a neighbour

Serial Solution

- Receptionist performs the following tasks
  1. Asks each student their assigned room number
  2. Works out their new room number
  3. Informs them of their new room number

Parallel Solution

“Everybody check your room number. Multiply it by 2 and go to that room”

Summary

- CUDA Programming Model
  - Present the processing flow for running a GPU program
  - Explain the CUDA software model and its relation to the hardware hierarchy
  - Propose a simple problem which can be implemented on the GPU

- Next Lecture: CUDA Device Code

Acknowledgements and Further Reading

- Some of the content in this lecture material has been provided by;

1. GPUComputing@Sheffield Introduction to CUDA Teaching Material
   - Originally from content provided by Alan Gray at EPCC/NVIDIA
2. NVIDIA Educational Material
   - Specifically Mark Harris's (Introduction to CUDA C)

- **Further Reading**
  - Essential Reading: CUDA C Programming Guide
    - http://docs.nvidia.com/cuda/cuda-c-programming-guide/