Workshop 2 – RESCUER MSCA DOCTORAL NETWORK 2024-2028
Universidad de Zaragoza

GPU programming (II)

Task 2: CUDA kernel implementation

Mario Morales Hernández mmorales@unizar.es

Module 4 – GPU programming (II)

Task 2 - CUDA kernel implementation

The second day focuses on implementing CUDA kernels for key functions in the shallow water equations solver. The goal is to convert CPU-based functions into efficient GPU kernels while ensuring correctness and performance. The first part will focus on flux computation functions (*h_compute_x_fluxes* and *h_compute_y_fluxes*), while the second part will address wet/dry cell checking (*h_wet_dry_x* and *h_wet_dry_y*) and managing host-device memory transfers for functions that remain on the CPU.

Key tasks:

1. Convert h_compute_x_fluxes and h_compute_y_fluxes:

    - Use the flattened loops from Task 1 to implement CUDA kernels.
    - Add the __global__ keyword to define the kernels. Use the same macro used in Task 1.
    - Implement thread indexing using blockIdx.x, threadIdx.x, and blockDim.x.
    - Implement the kernel call, paying attention to the grid dimension and blockSize.

2. Convert h_wet_dry_x and h_wet_dry_y:

    - Use the flattened loops from Task 1 to implement CUDA kernels.
    - Add the __global__ keyword to define the kernels. Use the same macro used in Task 1.
    - Implement thread indexing using blockIdx.x, threadIdx.x, and blockDim.x.
    - Implement the kernel call, paying attention to the grid dimension and blockSize.

3. Keep certain functions on the CPU: Leave h_compute_flow_time_step_2D and h_check_depth_positivity on the CPU.

    - Use cudaMemcpy to transfer data between the host and device as needed.
    - Implement proper memory transfers:
    - Identify variables required for CPU functions and transfer them efficiently between the host and device.

4. Modify swe2d.c to update the kernel calls for the converted subroutines (h_compute_x_fluxes, h_compute_y_fluxes, h_wet_dry_x, and h_wet_dry_y). Use the same macro pattern from Day 1 to keep the CPU and GPU versions clean and maintainable. Pay attention to the kernel call configuration, ensuring the correct grid and block dimensions. Use the nThreads macro for the block size.