Universidad Zaragoza
1542

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

GPU programming (III)

Task 3: Race condition detection and resolution

Mario Morales Hernández mmorales@unizar.es

Module 4 – GPU programming (III)

Task 3 - Race condition detection and resolution

The third day focuses on identifying and resolving race conditions in the CUDA implementation of the shallow water equations solver. Students will analyze the code to understand how race conditions arise, investigate the flux computation pattern, and develop solutions to ensure correctness.

Key tasks:

1. Run the code and analyze mass conservation:

    - Use the h_compute_water_mass function to monitor mass conservation.

    - Observe that mass is not perfectly conserved and identify discrepancies.

    - Investigate where variables (DU1, DU2, DU3) are being modified simultaneously by multiple threads.

2. Examine the flux computation pattern:

    - Analyze how fluxes are accumulated and which cells are affected by each interface computation.

    - Draw a diagram to visualize:

        - Two adjacent cells (left and right)

        - The threads that modify each cell's DU values.

        - The potential concurrent modifications.

3. Identify Race Conditions:

    - Answer the key question: "What happens when multiple threads try to update DU1, DU2, DU3 for the same cell?"

    - Determine which parts of the code are prone to race conditions.

4. Develop a solution to prevent race conditions:

    - Discuss strategies to separate updates and prevent simultaneous modifications.

    - Consider splitting the updates based on their direction (e.g., separate arrays for different update directions).

    - Implement proper memory allocation and management for the new arrays.

5. Validate the solution:

    - Compare mass conservation before and after implementing the fix.

    - Test the solution to ensure correctness.

    - Verify that the solution resolves the race conditions without introducing new issues.

6. Modify swe2d.c:

    - Update the kernel calls and memory management to incorporate the solution. Every place that has DU1, DU2, DU3 has to be replaced with the new variables.

    - Ensure that the code remains clean and maintainable.