Lab 2: 7-point Stencil with Thread-coarsening and Register Tiling

EE5355 - Parallel Algorithmic Techniques

Due: See Moodle

1 Objective

The purpose of this lab is to practice thread coarsening and register tiling optimization techniques using 7-point stencil computation as an example.

2 Procedure

Step 1: Transfer the tarfile lab2.tgz into your project directory and extract it to create the lab2 directory.

Step 2: Edit the `kernel` function in `kernel.cu` to implement a 7-point stencil with combined register tiling and x-y shared memory tiling, and thread coarsening along the z-dimension.

Step 3: Edit the `launchStencil` function in `kernel.cu` to launch the kernel you implemented. The function should launch 2D CUDA grid and blocks, where each thread is responsible for computing an entire column in the z-dimension.

Step 4: Compile and test your code:

```
make
./stencil                    # Cube size = default (512x512x64)
./stencil <NX> <NY> <NZ>     # Cube size = NX x NY x NZ
```

Step 5: Edit the file `report.txt` to answer the following questions:

1. Briefly describe your implementation in one or two paragraphs as well as any difficulties you faced in developing and optimizing the kernel.

2. Consider a 100x100x50 7-point stencil that does not use any thread-coarsening or tiling optimizations. How much real computation does each work-item perform? How many global memory loads does each work-item perform? What is the ratio of computation to global memory loads for each work-item? (Consider as useful work only additions, subtractions,

and multiplications that operate directly on the loaded data. Do not consider index computations.)

3. Consider a 100x100x50 7-point stencil that uses thread-coarsening and joint tiling optimizations such as the one you implemented. How much real computation does each work-item perform? How many global memory loads does each work-item perform? What is the ratio of computation to global memory loads for each work-item? (Consider as useful work only additions, subtractions, and multiplications that operate directly on the loaded data. Do not consider index computations.)

4. Briefly comment on the difference in computation to global memory load ratios for the two cases.

Step 6: Submit your assignment. You should only submit the following files as a tarfile via Moodle.

- `kernel.cu`
- `report.txt`

```
tar -zcvf lab2_submission.tgz kernel.cu report.txt
```

3 Grading

Your submission will be graded based on the following criteria:

- Functionality/knowledge: 75%
  - Thread-coarsening implemented correctly
  - Register tiling implemented correctly
  - Shared memory tiling implemented correctly
  - Performance
- Report: 25%
  - Questions answered correctly
  - Neatness and clarity