PAPM/OpenCL: Assignment 3

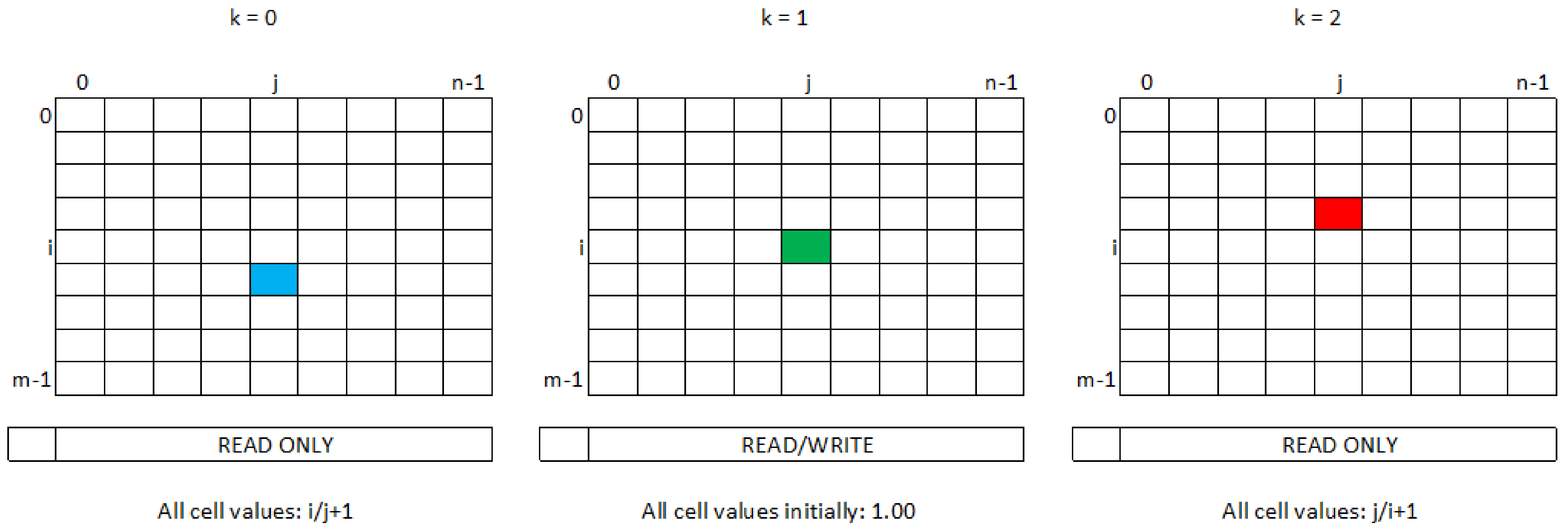
VALON RACA

Problem

Mathematical formulation

Consider:

- A three-dimensional *matrix A* with dimensions *m x n x p*, where m, n = 8192, p = 3, and indexes 0 ≤ i < m, 0 ≤ j < n, 0 ≤ k < p.
- Elements a_{ijk} have the following values:
  - $a_{ij0} = \frac{i}{j+1}$
  - $a_{ij1} = 1.00$
  - $a_{ij2} = \frac{j}{i+1}$



- An iterative process with 0 < t < 23 iterations shall evaluate the following equation, only for k=1, 1 <= i < m-1, 0 <= j < n:

$$a_{ij1}^{(t)} = a_{ij1}^{(t-1)} + \frac{1}{\sqrt{a_{(i+1)j0} + a_{(i-1)j2}}}$$

Tasks

1. Produce the serial version for comparison (in C/C++)
2. Develop an OpenCL application:
    1. You can simplify the problem: let the size of matrix be divisible by 32
    2. Produce a parallel version of the problem as an OpenCL kernel (must utilize multiple work-groups, each work-group must have multiple work-items)
    3. Integrate into application, which:
        1. Creates input of user-defined size divisible by 32 (filled by random values)
        2. Execute OpenCL kernel to produce output (using user-defined number of iterations)
        3. Check correctness of the output by comparison with the serial version
        4. Optionally print output from serial and parallel version
    4. Experiment with optimizations: use local memory, tune thread granularity (how much workload is done per thread), tune work-group size etc. (try to use at least two different optimizations)
        1. Test the performance on reasonably large input (e.g., matrix size 2048x2048 and up to 8192x8192)
3. Test the application on GPUs in ALMA compute nodes, your code must compile and run there

Deliverables

1. Files to be delivered
    1. The required application: one .cpp file or archive containing multiple source files with Makefile
    2. When optimizing the code, put the code into separate .cpp or archive
2. Submit a report, focused on the deliverables above
    1. Explain how your code is parallelized, describe used optimizations (also failed ones, which do not bring any speedup)
    2. You can explain also ideas for more complicated optimizations, even if you have not experimented with them (e.g., do you need to read the matrix from global memory for each iteration?, or how to utilize multiple GPUs?)

Machine to be used

- alma.par.univie.ac.at
- SSH access
  - you should have the ssh access to the machine
- Each alma node (alma01-alma06) has 2x NVIDIA Tesla K20
  - srun -N1 --pty bash -i
- Host code compilation
  - add #include “CL/cl.h” to host code
  - compile host code
    - g++ -O3 -I/usr/local/cuda-10.2/targets/x86_64-linux/include/ -L/usr/local/cuda-10.2/targets/x86_64-linux/lib/ -o my_project my_project.cpp -lOpenCL

Submission

1. Submit all files required in slide Deliverables
   1. In a .zip file (named: FirstName_LastName.zip)
   2. Include source files/archives and the report

2. Deadline
   1. Tuesday, 25.01.2022 23:55
   2. Via Moodle Platform

3. Contact in case of problems
   1. Via Moodle forums
   2. Via Email: valon.raca@univie.ac.at