COMP 429/529: Assignment 2

Due: 11.00 pm on Sat, April 25th, 2020

Notes: This assignment is worth 20% of your total grade, can be done as a team of 2 or alone. You may discuss the problems with your peers but the submitted work must be your own work. Post your project related questions to the Blackboard forum.

Corresponding TA: Palwisha Akhtar (pakhtar19@ku.edu.tr).
Her office hours: Thursdays 16:00-18:00 pm or by appointment

Note that this assignment will require a considerable amount of coding. So start early.

Accelerated Image Denoising

In this assignment, you will work on a noise reduction algorithm that removes noise from an image. The image denoising algorithm we are using is very simple compared to the block matching or Bilateral filters. The algorithm is a diffusion method typically used for ultrasonic and radar imaging applications based on partial differential equations (PDEs). It is used to remove locally correlated noise, known as speckles, without destroying important image features. The program consists of three stages in continuous iterations over the image (reduction/statistics, computation 1 and computation 2). The sequential dependency between all of these stages requires synchronization after each stage (because each stage operates on the entire image).

Serial Code

You are provided with the serial version of image denoising algorithm. The code has various options as follows:

```
 ./noise_remover

With the arguments
-i Input image to be denoised
-iter Number of iterations
-l lambda which is the step size
-o Output denoised image
Example command line
    ./noise_remover -i coffee.pmg -iter 100 -o denoised_coffee.png
```

You will parallelise the provided serial algorithm using CUDA. Starting with the serial implementation, the assignment requires 3 versions of the same code:

- Version 1: Parallelize the algorithm using a single GPU. First implement a naive version that makes all the references through global memory. Make sure that your naive version

works correctly before you implement the other options. Check if all the data allocations on the GPU, data transfers and synchronisations are implemented correctly.

- Version 2: Use temporary variables (in registers) to eliminate global memory references for arrays in the calculation. Take advantage of the data reuse.
- Version 3: Then develop a CUDA implementation by using shared memory (on-chip memory) on the GPU by bringing a 1D or 2D block into shared memory and sharing it with multiple threads in the same thread block. Perform this optimisation only to Compute 2 loop for grading (You can apply it to the others if you would like).
- You should implement these versions on top of each other (e.g. Version 3 should be implemented on top of Version 2).
- For details on how to implement these optimizations, refer to Lectures on GPU memory optimizations.
- Note that these optimisations do not guarantee performance improvement. Implement them and observe how they affect the performance.
- Optimize the code as much as you can. You can use techniques not mentioned here or not covered in the class. Document each of the optimizations in your report.

For full credit, you have to implement all these versions and submit the relevant codes. You can refer to the Nvidia Best Practices which discuss other types of optimisations:
http://docs.nvidia.com/cuda/cuda-c-best-practices-guide/index.html#abstract

Validation

Your parallel implementations should give the same result as the serial implementation. It might be difficult to inspect the images visually to check if the results are the same. The current program takes the average sum of the pixels in the final images. Use this number reported in the standard output as a way to check the program correctness. If the average does not match with the serial version's, then something in your program is probably wrong.

Environment

We will be using the KUACC cluster for this assignment.

- You need to add gres parameter to be able to use GPUs. You can
  refer to the example jobscript at /kuacc/jobscripts/mumax3/mumax3_submit.sh
- or on the web https://docs.computecanada.ca/wiki/Using_GPUs_with_Slurm
- For interactive GPU usage, you can use the following commands:

```
srun -A users -p short -n1 --gres=gpu:1 --qos=users --pty $SHELL

srun -A users -p short -n1 --gres=gpu:gtx_1080ti:1 --qos=users --pty $SHELL
```

- For batch jobs, you need to add the following parameter into your scripts

```
#SBATCH --gres=gpu:tesla_k20m:1
or
#SBATCH --gres=gpu:1
#SBATCH constraint=tesla_k20m
```

- If you have problems compiling or running jobs on KUACC, first check the website provided by the KU IT. If you cannot find the solution there, you can always post a question on Blackboard.
- Don't leave the experiments on KUACC to the last minutes of the deadline as the machine gets busy time to time. Note that there are many other people on campus using the cluster.

Reporting Performance

- Use the image files provided with the assignment for testing. Report the performance for the coffee.pgm image for iter=100.
- We have provided you the code to measure the Gflop/s rates. Note that at each iteration, for each data point, we perform about 45 floating point operations (divisions, multiplies and adds). Then you can compute the flop rates as follows:

```
int num_flops_per_point = 45 ; // see the exact formula in the code, it is not exactly 45

double flop_rate = num_iterations *
     (1E-9 * grid_rows * grid_cols * num_flops_per_point) / total_time;
```

- Tune the block size for your implementation and draw a performance chart for various block sizes for one of the images.
- Compare the performance of your best implementation with the CPU version (serial).
- Compare the performance of different CUDA versions.
- Document these in your report.

Submission

- Submit all three versions of the code, label them properly. If you have another version that implements different optimizations, submit that as well.

- The first paragraph of your report should clearly state which versions work properly and which version is the best performant one.
- In the first paragraph explicitly state the Gflops rate you achieve for coffee.pgm and iter=100.
- Document your work in a well-written report which discusses your findings. Offer insight into your results and plots.
- Submit both the report and source codes electronically through blackboard.
- Please create a parent folder named after your username(s). Your parent folder should include a report in pdf and a subdirectory for the sources. Include all the necessary files to compile your code. Be sure to delete all object and executable files before creating a zip file.

Grading

Your grade will depend on 3 factors: performance, correctness and the depth and your explanations of observed performance in your report.

Implementation (80 points): Version 1 (30 pts), Version 2 (20 pts each), Version 3 (30 points)

Report (20 points): implementation description and any tuning/optimisation you performed (10 pts), performance study (10 pts).

GOOD LUCK.