COMPUTAÇÃO DE ALTO DESEMPENHO

2022/2023
Project 1 – CUDA
Deadline: ~~14~~ 20 November 2022

The heat diffusion over some surface (2D) over time can be described by a partial differential equation given by:

$$\frac{\partial T}{\partial t} = \alpha\left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2}\right)$$

Where α is heat diffusion coefficient, T is temperature at point (*x,y*) over time *t* .

An approximation can be achieved by numerical methods that discretizes the equation, replacing the derivatives with infinitesimal differences over x, y and t. So, the temperature variation at point (x,y) for some time step (*dt*) can be approximated by the contributions of the neighboring points. Using the following approximation for small *dx* and *dy*:

$$\frac{\partial^2 T}{\partial x^2} \approx \frac{T(x-dx) - 2T(x) + T(x+dx)}{dx^2}$$

the new temperature after some small *dt* at some point (x,y), for *dx=dy=h*, is given by:

$$T_{x,y}^{t+dt} = T_{x,y}^{t} + \frac{dt\alpha}{h^2}\left(T_{x-1,y}^{t} + T_{x+1,y}^{t} + T_{x,y-1}^{t} + T_{x,y+1}^{t} - 4T_{x,y}^{t}\right)$$

This solution is stable and a good approximation for:

$$dt \le \frac{h^2}{4\alpha}$$

So, to simulate areas or with more resolution, a smaller *dt* must be used and so, more timesteps are needed to compute the temperatures for the same time span or for longer time spans.

A sequential code example is provided for reference, that uses a grid of points that start at some initial state, progressing over time in steps and calculating the new grid states. The time interval *dt*, used in these steps, are computed from the previous stability condition. To compute the surface state at some instant *t*, the number of steps must be increased to reach that time (*numSteps=t/dt*).

The main objective is to achieve the best performance, particularly for bigger resolutions and computed time spans. For convenience, this work can be divided in several stages (number 5 is mandatory):

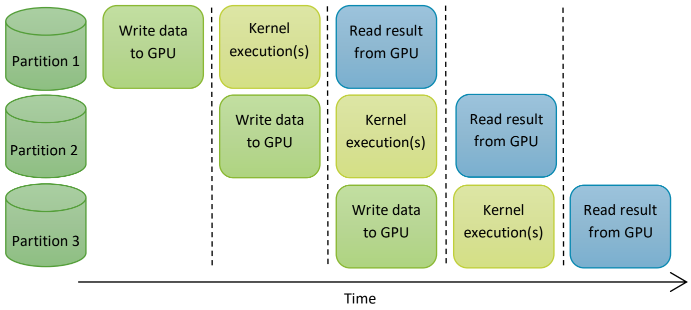
1. (30%) Implement a CUDA or OpenCL solution that
    a. Parallelizes the heat diffusion computation
    b. Experiment with different parallel strategies. Examples: different grid layouts, block sizes, number of kernels executed, work size done per kernel and work done per thread.
    c. Study if using a local shared area can improve your kernel(s) performance (nvprof and section 8 of CUDA Best Practices Guide[1] may help you evaluate any improvement).

2. (30%) Evaluate these solutions
    a. against the sequential version
    b. using different grid arrangements and thread block sizes
    c. using and not using shared memory

    Note1: ignore file I/O times but include in your timings memcopy to/from device.
    Note2: use a relevant workload. Try computing with parameters: nx=ny=200, h=0.005, numSteps=100000; with output image every 10000 steps or just at the end.

[1] https://docs.nvidia.com/cuda/cuda-c-best-practices-guide/index.html#performance-metrics

3. (10%) Complement your solution with the ability to overlap computation with communication using CUDA streams. Consider the following example with 3 streams.



While kernel(s) is(are) being applied over Partition 2, Partition 3 can be simultaneously uploaded to the GPU and, possibly, the result of processing Partition 1 can be simultaneously downloaded from the GPU. You must define what are your partitions in this problem.

4. (10%) Evaluate this solution against the others as in stage 2.

5. (20%) Write a report (max of 5 pages A4 11pt font) that presents
   a. Tested approaches and final (best) solution
   b. Relevant implementation details
   c. Your evaluation results (include times and/or graphs to compare and justify your solution)
   d. An analysis and interpretation of all these results

The report, in pdf format, and final solution source code, should also be delivered in a zip file by email.

Other relevant optimizations may also be accounted in the final grade.

Final notes:
This work is based on others (including solutions) available on the internet, like:
https://github.com/csc-training/openacc/tree/master/exercises/heat
https://enccs.github.io/OpenACC-CUDA-beginners/2.02_cuda-heat-equation/