Runtime comparison solving Gray-Scott equation on different OpenCL devices

Michael Quell
michael.quell@yahoo.de

TECHNISCHE UNIVERSITÄT WIEN

Abstract

An example of a reaction-diffusion equation with chaotic solutions. You can expect patterns to emerge from chaos. A uniformly discretization in space and periodic boundary conditions allows the Fast Fourier Transform to be used, so that when coupled with a suitable time stepping scheme a numerical method that suits the parallelism of OpenCL is obtained. The code was benchmarked on various CPU and GPU devices. Performance results for various problem sizes are shown. Example code can be found at:
`https://github.com/MichaelQuell/GrayScott-OpenCl`

The Equation

The Gray-Scott-equation[1] describes the reaction of 2 chemicals and it is given by

$$\frac{\partial u}{\partial t} = D_u \Delta u - uv^2 + F(1-u),$$
$$\frac{\partial v}{\partial t} = D_v \Delta v + uv^2 - (F+k)v.$$

The equation is solved on $[-4\pi, 4\pi]^2$ with periodic boundary conditions. The parameters used in the simulations were $D_u = 0.04$, $D_v = 0.005$, $F = 0.038$ and $k = 0.076$. The initial data is a 2-dimensional Gaussian function in both components.

Solving

The equation is solved using a splitting method, in this case we split the equation into the non linear

$$\frac{\partial u}{\partial t} = -uv^2, \quad \frac{\partial v}{\partial t} = uv^2$$

and the linear part

$$\frac{\partial u}{\partial t} = D_u \Delta u + F(1-u),$$
$$\frac{\partial v}{\partial t} = D_v \Delta v - (F+k)v.$$

The linear part can be solved exactly in Fourierspace and the non linear could be exactly solved using the Lambert W function, but there is no implementation available for OpenCl, instead fix point iteration is used to solve the implicit midpoint rule. The two parts are combined with Strang splitting[2]. The space discretization is equidistant in both axes.

Implementation

1. Initialize the data
2. Time stepping
   (a) Call non linear kernel
   (b) **Compute FFT** [3]
   (c) Call linear kernel
   (d) **Compute iFFT**
   (e) Call non linear kernel
   (f) Do some output
3. Shut down the program

The linear and non linear kernel operates on each grid point individually and are perfectly suit the parallelism of OpenCl. The most time consuming step is the computation of the FFT and iFFT. Also there is only data transfers from the GPU, when you do some output.

Results

The codes have been tested on following systems, all running Ubuntu 14.04:

sys 1) AMD-A10-5800K (4×3.8) GHz, APU Radeon HD 7660D, 8Gb ram

sys 2) Intel i7-4700MQ (4×2.4) GHz, GeForce GT 755m, 16Gb ram

sys 3) AMD FX-8350 (8×4.0) GHz, Radeon HD 5450, 4Gb ram

For the performance measuring 500 steps are computed and the Wall-clock time is used. The problem size started at 32^2 and is doubled every time, till 4096^2 is reached for single precision and 2048^2 for double precision, because of the memory limitation on the GPU's.

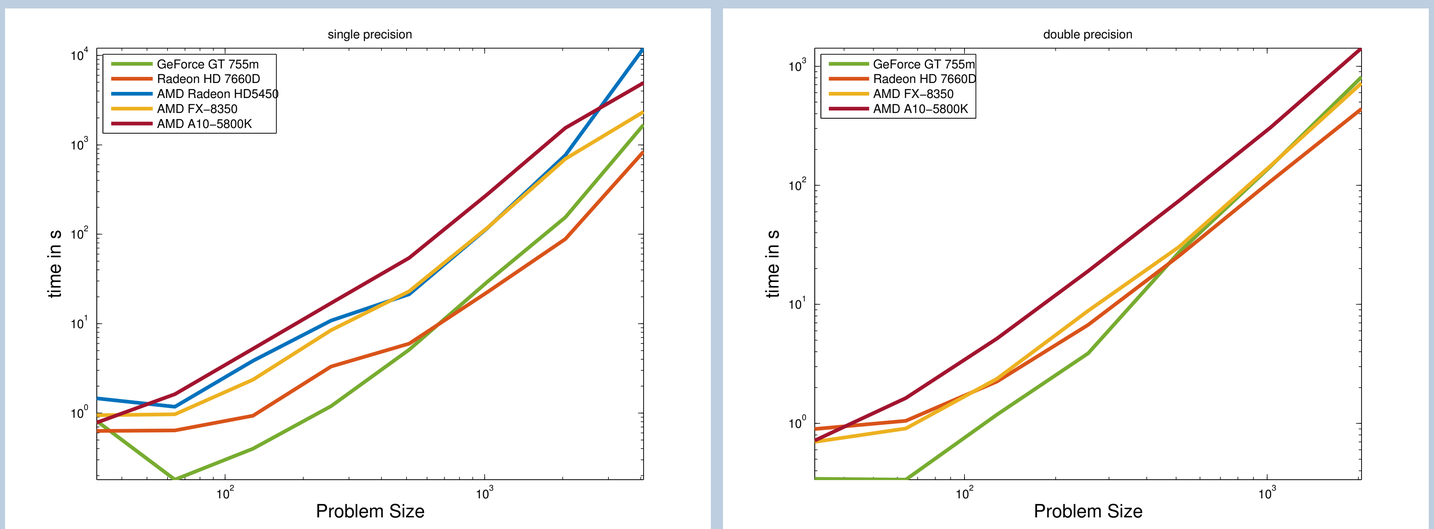

Figure 1: Calculation with single precision (left) and double precision (right)

Figure 2: Left to right $t = 0, 500, 1500$

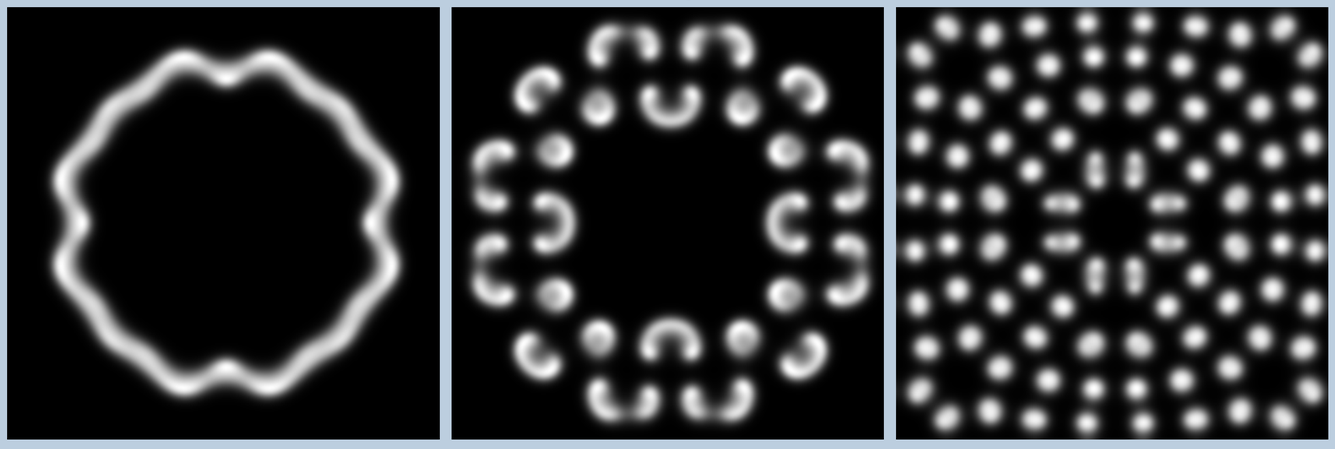

Figure 3: Left to right $t = 2250, 2500, 3000$

Conclusion

The results show that the GPU's outperform the CPU's in single precision, but when computing double precision the performance of the CPU's stays the same, while the GPU's have a significant drop. To compute the equation on larger grids or in 3-dimension, you will, have to face the problem that on GPU's memory is short. One could avoid that by writing a distributed FFT as the other kernels are independent on the problem size.

References

[1] P. Gray, S. Scott, Chemical Waves and Instabilities, Clarendon, Oxford, 1990.
[2] G. Strang, On the construction and comparison of difference schemes, SIAM J. Numer.Anal., 5:506-517, 1968.
[3] clFFT open source library to compute FFT with OpenCl `https://github.com/clMathLibraries/clFFT`

Acknowledgements

I would like to thank Benson K. Muite for funding the hardware for the research.