

Runtime comparison solving Gray-Scott equation on different OpenCL devices

Michael Quell
Institute of Analysis and Scientific Computing
Wiedner Hauptstraße 8-10
Vienna, Austria
michael.quell@yahoo.de

ABSTRACT

n 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>

CCS Concepts

•**Computing methodologies** → *Massively parallel and high-performance simulations*; •**Applied computing** → *Chemistry*;

Keywords

Gray-Scott, Splitting, OpenCL, FFT

1. INTRODUCTION

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

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

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.

2. SOLVING

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

IWOCL '16 April 19-21, 2016, Vienna, Austria

© 2016 Copyright held by the owner/author(s).

ACM ISBN 978-1-4503-4338-1/16/04.

DOI: <http://dx.doi.org/10.1145/2909437.2909461>

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

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

The linear part can be solved exactly in Fourier space 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.

2.1 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 suited to 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.

3. CONCLUSIONS

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.

4. ACKNOWLEDGEMENTS

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

5. 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>