Technical University of Denmark

Wojciech Pawlak (s091820)

Final Report

A GPU-accelerated Navier-Stokes Solver using OpenCL

Special Course at GPUlab, Scientific Computing Section, DTU
Supervisor: Allan P. Engsig-Karup Ph.D.

July 22, 2012

DTU Informatics
Department of Informatics and Mathematical Modelling

Contents

1 Introduction

The goal of this report is to document the work done for individual special course "A GPU-accelerated Navier-Stokes Solver using OpenCL". The project took place in GPUlab of Scientific Computing Section of Department of Informatics and Mathematical Modeling (DTU Informatics) at Technical University of Denmark. The project work for this special course took place between late March until early August 2012 and was supervised by Allan P. Engsig-Karup Ph.D. The course was worth 5 ECTS points.

The practical goal of this project was to design and implement a scientific computing application for execution on GPUs. The implemented application was then verified and validated using standard benchmarks.

The performance of implemented PDE solver was analyzed.

Apply optimization techniques for improving performance of PDE solver on GPUs.

Understand how to write a parallel program using OpenCL for heterogenous computing on many-core architectures.

Apply basic principles for numerical approximation/discretization.

Section 2 is

Section 3 is

Section 4 is

Section 5 is

2
Survey of GPU programming

AMD APP Acceleration

http://stackoverflow.com/questions/1126989/what-future-does-the-gpu-have-in-computing

2.1 CUDA/OpenCL

2.2 Previous research

2.3 Recent developments in CUDA and OpenCL

2.3.1 OpenCL

What applications use OpenCL GPU-acceleration? OpenCL in Photoshop CS6 WinZip 16.5

OpenCL Studio

http://www.youtube.com/user/OpenCLStudio

2.3.2 CUDA

2.3.3 Other Technologies

C++ AMP (C++ Accelerated Massive Parallelism)

http://msdn.microsoft.com/en-us/library/hh265137

OpenACC

Directives like OpenMP for multicore CPU programming

http://openacc.org/

AMD Accelerated Parallel Processing (APP) SDK

formerly ATI Stream

http://developer.amd.com/sdks/AMDAPPSDK/Pages/default.aspx

http://www.amd.com/us/products/technologies/amd-app/Pages/eyespeed.aspx

2.4 Current advances in NVIDIA and AMD Architectures

2.4.1 NVIDIA

Tesla

Kepler

2.4.2 AMD

3 Navier-Stokes

3.1 Description of Navier-Stokes

Reynolds number < 1000

3.2 Implementation

3.2.1 Sequential

3.2.2 Parelleized on GPU

4 Design and Implementation

4.1 Design

A set of kernels

Modularity

All code should be executed on GPU for valid comparison.

4.2 Implementation

Code in project is based on structure from Griebels code. Least as possible was changed in structure as to keep it comparable with the original code.

Code is written in C and OpenCL C.

Two-dimensional arrays in original One-dimensional arrays for OpenCL code. 1D arrays cannot be passed to OpenCL kernels.

Worksize

Implementation consists of sets of kernels as

Griebels cod intrinsics: Memory allocation

Code is portable and works under both on Windows and Unix platforms. Tested on Windows 7 and Linux clusters. Compilers used are nvcc and Visual Studio on Windows and gcc on Linux.

Table Computer specs.

Table

4.2.1 Benchmarking code

Code is timed with standard library time.h clock function. No OpenCL timer code because on CPU. Code timed with buffer allocation/memory access and without.

4.2.2 Visualisation code

Matlab scripts to visualise code. Tests.

4.2.3 Naive kernels

Straightforward port of functions to kernels. Getting rid of for loops. Ensuring that boundaries are not crossed.

4.2.4 Shared memory kernels

5 Performance Analysis

5.1 Computed results

6 Conclusions

References