Particle-In-Cell Codes in CUDA

Olav Emil Eiksund

November 22, 2014

Abstract

abstract content

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

1.2 Terminology

CUDA

cuFFT

GPGPU

GPU

1.3 Thesis Outline

Briefly describe the contents of each section.

1.3.1 Introduction

1.3.2 Background

1.3.3 Implementation

1.3.4 Results

1.3.5 Discussion

1.3.6 Conclusion

Chapter 2

Background

2.1 Particle-In-Cell Codes

A technique for solving certian PDEs. Usually involves simulating particles or fluid in a grid. Also for plasma simulations.

Applications Used for studying plasma, electrons, ions. Other applications in solid/fluid mechanics.

2.1.1 Physics

This section should describe the physical basis for the simulation.

2.1.1.1 Subject of simulation

What is being modeled? A set of particles with position velocity mass and charge. Attempting to determine behaviour over time. Need to find force on particles. Need to determine electric field from potential. Find potential from distribution of charge from particles. Repeat. Simple.

Discrete in space and time. Details below.

2.1.1.2 Discrete Model Equations

Mention that these are the equations used in Elsters phd...

$$\nabla^2 \Phi = -\frac{\rho}{\epsilon_0} \tag{2.1}$$

$$E = -\nabla \Phi \tag{2.2}$$

$$\frac{dx}{dt} = v \tag{2.3}$$

$$\frac{dv}{dt} = \frac{qE}{m} \tag{2.4}$$

Poissons equation

$$\nabla^2\Phi = -\frac{\rho}{\epsilon_0}$$

$$\frac{\partial^2\Phi}{\partial^2 x} + \frac{\partial^2\Phi}{\partial^2 y} + \frac{\partial^2\Phi}{\partial^2 z} = -\frac{\rho}{\epsilon_0}$$

Poissons equation: a PDE that relates the charge to the Laplacian of the potential.

Describe discretization implications

Electric field from Potential

$$E = -\nabla\Phi$$

Relates the electric field strength to the potential. The electric field strength at a location is proportional to the gradient of the potential, and is sampled at each grid vertex by taking the difference in each direction.

Electric force from the electric field

$$F_{\text{electric}} = qE$$

Electrical force is defined as the field strength E times the charge q.

Acceleration, velocity - Electric force

$$\frac{dv}{dt} = \frac{qE}{m}$$

$$\frac{dx}{dt} = v$$

Knowing that the acceleration of a particle $a = \frac{F}{m} = \frac{qE}{m}$, we can estimate the velocity and position as $v = v_0 + a \cdot \Delta t$, $p = p_0 + v \cdot \Delta t$.

Describe discretization (time, space)

2.1.1.3 Procedure

1. Find charge density.
2. Solve for potential.
3. Determine electric field from potential.
4. Update particles.

2.2 Solvers

2.2.1 Boundary conditions

Outline which different boundary conditions can appear, and for which ones the different solvers are suited.

2.2.2 FFT

Short description of the concept behind the FFT.

2.2.2.1 Fourier Transform

Brief explanation of the math behind the fourier transform.

2.2.2.2 DFT

How the continous transform translates to discrete applications.

2.2.2.3 Cooley-Tukey FFT algorithm

On the C-T FFT.

2.2.2.4 Performance

NlogN complexity, etc.

2.2.2.5 Properties

Aliasing etc.

2.2.2.6 Applications

Some common applications of the FFT, and reasons it is used.

2.2.3 SOR

2.2.3.1 System of Linear Equations

Properties of a system of linear equations. Also how this problem is represented as one.

2.2.3.2 Gauss-Seidel

Efficient serial solver. How it translates to paralell.

2.2.3.3 Over-relaxation

How this works, and how it improves performance.

2.2.3.4 Performance

Some statistics to compare SOR to other solvers, reference point for comparison.

2.2.3.5 Properties

Tendency to ”float” under certain boundary conditions, inconvergence etc.

2.2.3.6 Applications

Examples of successful application of SOR as a solver, and reasons it was used.

2.3 Parallel Computing

This section explains the motivations behind parallel programing and GPGPU in general. It should show general characteristics of the CUDA architecture and programming model, and show the contrast to CPU programming, mentioning both the benefits and trade-offs, and reasons it is different.

2.3.1 Parallel computing basics

Briefly explain the essence of TDT4200, as much as is necessary to understand the rest of the report.

Moore's law Outline regular increase in computing power as transistor density decreased.

Power wall, etc Explain why new measures had to be taken to increase performance further, leading to increasing focus on multicore.

Amdahls law, Gustafsons law Explain what Amdahl and gustafsons laws mean for paralellization of serial programs. How to bypass the problems by inccreasing problem size.

2.3.2 GPGPU

Brief history of GPGPU, mention relationship to gustafsons law (increase problem size at cost of processing speed). Architechture and language agnostic, what are the benefits and costs of this approach.

2.3.2.1 History

How did GPGPU come about, and how has it developed since. Status today.

2.3.2.2 Purpose

Mathematical/scientific reasons that GPGPU performs well, and why it is used.

2.3.3 CUDA

Specifics on the cuda preogramming language/architecture. It's origin and development since.

2.3.3.1 Programming model

How to write a CUDA kernel. Why one writes a kernel. How a kernel differs from any other funciton call.

2.3.3.2 Architecture

The CUDA architecture and how it must be taken into consideration when writing CUDA programs.

General Genral layout, function and performance of the CUDA architecture.

Maxwell Specifics for the Kepler and Maxwell generations.

2.3.3.3 Memory

Memory management. Different kinds of memory, allocating memory, atomic operations etc...

2.3.3.4 cuFFT

Details on the cuFFT framework. How it is used, performance compared to FFTW...

2.4 Previous work

Chapter 3

Implementation

3.1 Architecture

Describe the structure and idea behind the program written.

3.1.1 Simulation

Explain how the simulation works,

3.1.2 Kernels

Description of all kernels and any special handling needed.

3.1.2.1 determineChargesFromPotential

The kernel determining the charge density of the mesh from particle charges.

3.1.2.2 electricFieldFromPotential

The kernel finding the electric field at a position given the potential.

3.1.2.3 updateParticles

The kernel updating position and velocity of the particles.

electricFieldAtPoint Helper function that determines the electric field at any position.

3.1.3 cuFFT

Decribe how cuFFT setup is performed, and how the FFT is run.

solve The kernel solving the potential in frequency space.

3.1.4 SOR

Explain how the SOR is performed.

SOR kernel Describe the SOR kernel itself.

3.1.5 Setup

Show how setup is performed: memory allocation, constants, parameters.

3.2 Performance

Uncertain what I intended to put here. Perhaps description of steps taken to improve performance, such as shared memory, aligned memory allocations (pitched pointers), perhaps data types, and any other optimizations...

3.2.1 Data

3.2.2 Solvers

3.2.2.1 FFT

3.2.2.2 SOR

3.2.3 Particle tracing

Brief description of how tracing is performed, and how it will affect performance? Perhaps also any ideas for how this can be improved in the future.

Chapter 4

Results

Uncertain whether this section is needed at all. Perhaps some benchmarking results...

4.1 Section 1

4.2 Section 2

4.3 Section 3

Chapter 5

Discussion

Ideas for discussion:

- whether PIC is suitable for GPGPU,
- whether CUDA is well suited,
- which solvers performs better,
- next idea...

5.1 Section 1

5.2 Section 2

5.3 Section 3

Chapter 6

Conclusion

Unsure what to conclude with, but let's just write the report first. Probably will contain a summary of the answer to questions from discussion. Should probably mention something that demonstrates how this is a suited project, and some ideas for wht to center the thesis around.

6.1 Future Work

Specifics on improvements to consider.