MIT Department of Nuclear Engineering

Thesis Prospectus

For the Degree of

Master of Science

Joshua Payne

Implementation and Performance Evaluation of a GPU Particle-in-Cell Code

Approved by:____________________________
Thesis Supervisor

Approved by:____________________________
Thesis Reader

Author:________________________________

Date:__________________________________

April 25, 2012

1 Introduction

Simulating plasma behavior can be incredibly difficult. The equations that govern plasma behavior are incredibly non-linear due to significance of self forces. One of the best ways to simulate plasma behavior is by simulating the behavior of individual particles. Tracking the movements and interactions of a small fraction of the $10^{20}/m^3+$ particles can provide a very accurate representation of the bulk behavior of a real plasma. This is the primary purpose of the particle in cell (PIC) method. PIC methods employ mixed particle tracking and field solving in order to solve for self consistent plasma behavior. The general PIC method is as follows;

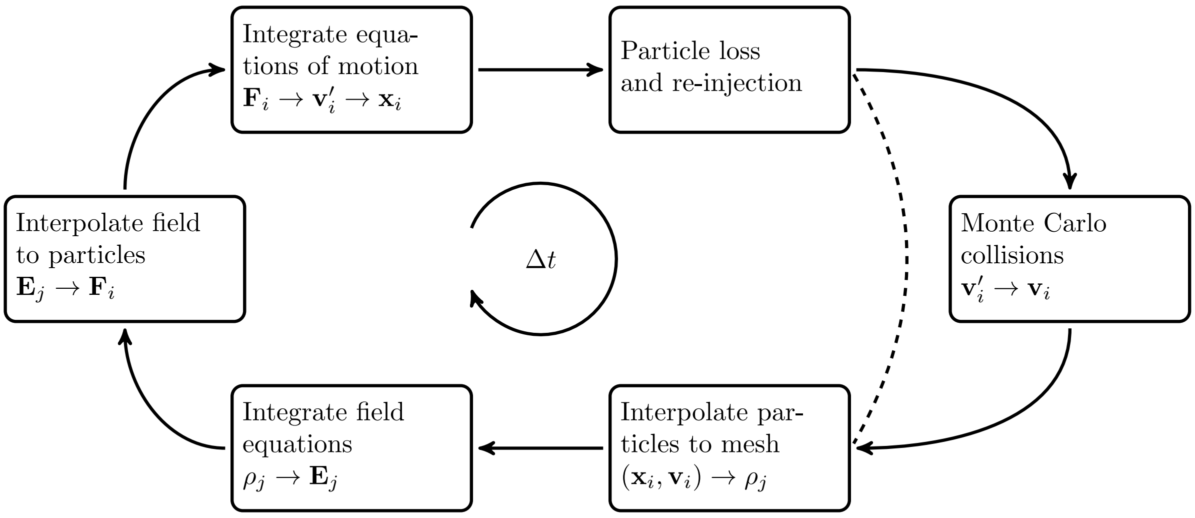
Integrate equations of motion $\mathbf{F}_i \to \mathbf{v}'_i \to \mathbf{x}_i$
Particle loss and re-injection
Interpolate field to particles $\mathbf{E}_j \to \mathbf{F}_i$
Δt
Monte Carlo collisions $\mathbf{v}'_i \to \mathbf{v}_i$
Integrate field equations $\rho_j \to \mathbf{E}_j$
Interpolate particles to mesh $(\mathbf{x}_i, \mathbf{v}_i) \to \rho_j$

Figure 1: Flow chart representation of the particle in cell method.

The problem with particle tracking codes is that they require a very large number of particles in order to achieve a reasonable accuracy on the order of tens of millions for small simulations up to tens of billions for large. The large number of particles required for good statistics means that PIC codes can be very slow. One way to reduce the computation time is to distribute the particle tracking across multiple processors. The ideal architecture for these codes would have a very large number of simple processors with very fast communication.

Graphical processing units or GPUs are designed to trace rays of light and generate an image. Ray tracing and particle tracking are very similar and because of this, GPUs have the potential to make great particle tracking processors. However, there are some issues with moving to a new architecture. The sheer amount of processing power reduces the time of many computations to the point where moving the data to and from the processor is more expensive than the computation. The lack of a large cache means that data access patterns and organization are significantly more important. Coupling GPUs and MPI introduces multiple levels of parallelism that have very different characteristics.

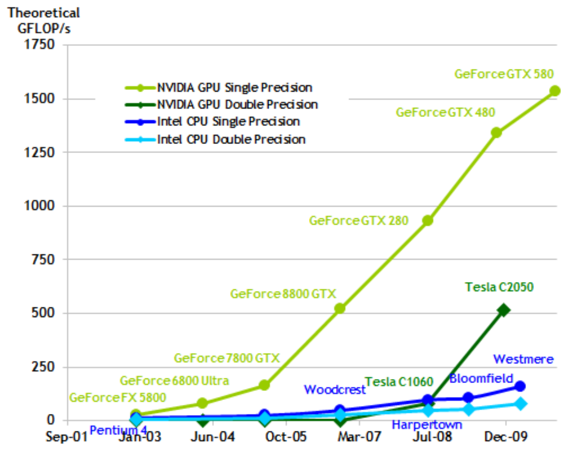

Figure 2: Performance comparison of GPUs vs CPUs.

The key to developing a high performance particle tracking code is properly decomposing and organizing the problem at each level of the multi-parallel tree. The complexity of the particle mover, collision operator, and timescale of the problem play large roles in how the problem is decomposed. This project will focus on development and performance characterization of generalized mulit-GPU PIC implementation techniques.

2 Background

Utilizing GPUs for PIC codes is a relatively recent development which coincides with the accessibility of general purpose gpu (GPGPU) computing. GPGPU computing has evolved rapidly since NVIDIA released the Compute Unified Architecture (CUDA) in 2008. Parallelization of the particle advancing step has been relatively straightforward, however, the developing an efficient charge assign on the GPU is more difficult, unless the particles are sorted.

There are four main approaches to maintaining a sorted particle list, the particle linked list by Heiko Burau et al [3], particle Quicksort by George Stantchev et al [13], and particle passing by Xianglong Kong et al [10], and the full particle sort using a radix sort. Each of these approaches has advantages and disadvantages, and will be characterized based on performance, ease of implementation, and generality. Additional work in the area of GPU PIC codes can be seen in [1, 2, 4, 9].

3 Objectives

The goal of this project is to investigate, develop, and characterize a GPU implementation of the PIC method. The code that will serve as the basis for this project is sceptic3D, a 3D electrostatic PIC code used to model ion collection by a sphere in a flowing plasma, further described in [5–8, 11, 12]. GPU versions of specific subcomponents of sceptic3D will be implemented using CUDA C, and compared to the original CPU code. Various GPU-PIC implementation techniques will be explored and evaluated based on their performance, ease of implementation, and applicability to a generalized PIC problem. The thesis will detail the development process and the reasoning behind choosing a specific technique for the final sceptic3Dgpu implementation.

4 Schedule

September 2011:

- Profile sceptic3D and develop Fortran-CUDA api.
- Compare particle structures, Structure of Arrays vs Array of structures.
- Redesign re-injection technique to be compatible with GPU version.

October 2011:

- Research and test particle sorting methods.
- Implement, debug, and test GPU versions of ptomesh and charge assign subroutines.

November 2011: Develop and implement GPU particle advancing subroutine.

December 2011:

- Enable multi-gpu capabilities.
- Begin GPU implementation of field solve subroutine.

January 2012:

- Finish GPU field solve, clean up Makefile.
- Perform basic optimizations.
- Begin benchmarking code. Outline thesis.
- Begin Introduction and Implementation chapters.

February 2012:

- Finish performance benchmarks, prepare performance figures.
- Start performance chapter.

March 2012: Complete drafts of Design and Implementation chapters.

April 2011: Complete rough draft of thesis.

May 2012: Revise and submit final draft of thesis.

References

[1] Paulo Abreu, Ricardo a. Fonseca, João M. Pereira, and Luís O. Silva. PIC Codes in New Processors: A Full Relativisitic PIC Code in CUDA-Enabled Hardware

With Direct Visualization. *IEEE Transactions on Plasma Science*, 39(2):675–685, 2011.

[2] Dominique Aubert, Mehdi Amini, and Romaric David. A Particle-Mesh Integrator for Galactic Dynamics Powered by GPGPUs. *Proceedings of the 9th International Conference on Computational Science*, 55444:874–883, 2008.

[3] Heiko Burau, Renée Widera, Wolfgang Hönig, Guido Juckeland, Alexander Debus, Thomas Kluge, Ulrich Schramm, Tomas E Cowan, Roland Sauerbrey, and Michael Bussmann. PIConGPU : A Fully Relativistic Particle-in-Cell Code for a GPU Cluster. *October*, 38(10):2831–2839, 2010.

[4] Viktor K. Decyk and Tajendra V. Singh. Adaptable Particle-in-Cell algorithms for graphical processing units. *Computer Physics Communications*, 182(3):641–648, March 2011.

[5] I H Hutchinson. Ion collection by a sphere in a flowing plasma: 3. Floating potential and drag force. *Plasma Physics and Controlled Fusion*, 47(1):71–87, January 2005.

[6] I H Hutchinson. Collisionless ion drag force on a spherical grain. *Plasma Physics and Controlled Fusion*, 48(2):185–202, February 2006.

[7] IH Hutchinson. Ion collection by a sphere in a flowing plasma: I. Quasineutral. *Plasma physics and controlled fusion*, 1953, 2002.

[8] IH Hutchinson. Ion collection by a sphere in a flowing plasma: 2. Non-zero Debye length. *Plasma physics and controlled fusion*, 1477, 2003.

[9] Rejith George Joseph, Girish Ravunnikutty, Sanjay Ranka, Eduardo D'Azevedo, and Scott Klasky. Efficient GPU Implementation for Particle in Cell Algorithm. *2011 IEEE International Parallel & Distributed Processing Symposium*, pages 395–406, May 2011.

[10] Xianglong Kong, Michael C. Huang, Chuang Ren, and Viktor K. Decyk. Particle-in-cell simulations with charge-conserving current deposition on graphic processing units. *Journal of Computational Physics*, 230(4):1676–1685, February 2011.

[11] L. Patacchini. *Collisionless Ion Collection by a Sphere in a Weakly Magnetized Plasma by Leonardo Patacchini.* PhD thesis, Massachusetts Institute of Technology, 2007.

[12] L. Patacchini. *Collisionless ion collection by non-emitting spherical bodies in E x B fields.* PhD thesis, Massachusetts Institute of Technology, 2010.

[13] G Stantchev, W Dorland, and N Gumerov. Fast parallel Particle-To-Grid interpolation for plasma PIC simulations on the GPU. *Journal of Parallel and Distributed Computing*, 68(10):1339–1349, October 2008.