Contents lists available at SciVerse ScienceDirect

Journal of Computational Science

journal homepage: www.elsevier.com/locate/jocs

Particle-in-cell plasma simulation on heterogeneous cluster systems

S. Bastrakov[a], R. Donchenko[a], A. Gonoskov[b], E. Efimenko[b], A. Malyshev[a], I. Meyerov[a,*], I. Surmin[a]

[a] *Lobachevsky State University of Nizhni Novgorod, Russia*
[b] *Institute of Applied Physics, Russian Academy of Sciences, Nizhni Novgorod, Russia*

ARTICLE INFO

Article history:
Received 5 April 2012
Received in revised form 19 August 2012
Accepted 22 August 2012
Available online 29 August 2012

Keywords:
PIC code
Large-scale simulation
Scalabilty analysis
GPU optimization
Supercell

ABSTRACT

This paper offers an approach to high-performance implementation of the Particle-in-Cell (PIC) algorithm for GPU-enabled heterogeneous cluster systems. We present Picador — a tool for three dimensional plasma simulation based on the fully relativistic PIC algorithm aimed at research of high intensity laser-matter interaction. It scales up to at least 2048 CPU cores with 85% strong scaling efficiency and 512 GPUs with 64% weak scaling efficiency, achieving up to 28% of the peak performance on the CPU and 14% on the GPU, respectively.

1. Introduction

Simulation of plasma dynamics and interaction of high intensity laser pulses with various targets in particular is one of the currently high-demand areas of computational physics. Among possible applications are creation of compact sources for hadron therapy for treating oncological diseases, creation of short-lived isotope factories for bioimaging, and development of tools for research on intramolecular and intraatomic processes. Solving practical problems is often possible only on supercomputers — problems are known that require simulation of the dynamics of $\sim 10^9$ and more particles in an area represented by $\sim 10^8$ grid cells.

Due to the high degree of nonlinearity and geometric complexity of the problem, plasma dynamics research is often based on simulation with the particle-in-cell (PIC) algorithm [1]. In the most general case it implies numerical solving the equations of plasma particles motion together with the Maxwell's equations on a discrete grid. Since the development in 1950s the method has been widely used in applied and fundamental research in many areas including astrophysics, high intensity laser interaction science, inertial confinement fusion and tokomaks issues. Among the advantages of the algorithm are its intuitive clarity, straightforward relation to the classical plasma physics and computational efficiency.

Nowadays, the development of the PIC method mainly continues in three directions. The first one is development of numerical methods reducing computational errors of existing algorithms, which is usually related to charge and energy conservation, numerical heating, numerical dispersion, numerical instabilities, etc (see Refs. [2,3] and references therein). The second direction is the modification of the well-known PIC algorithm to account for specific physical phenomena such as particle collisions, ionization, radiation reaction, quantum effects or to couple the existing code with different approaches such as fluid or kinetic methods to reach applicability and multi-scale properties in the context of the particular studies. This can be usually done in a straightforward manner due to the easy interpretation of PIC method in terms of classical physics approach. Finally, the third direction can be attributed to high-performance implementation of the PIC algorithm. During the last years there is growing interest in such research for both traditional cluster systems and heterogeneous systems offering graphics processing units (GPUs) as a source of immense computational power. The general PIC method includes interpolation of electromagnetic field for each particle and accounting for particle contribution to the current distribution. Both of these steps may involve lots of disordered memory accesses diminishing performance on GPUs. A promising approach is maintaining specific ordering of the particles data in GPU memory, e. g. using supercells in Ref. [4]. While both traditional supercomputer codes [5–12] and GPU codes [4,13] are being developed simultaneously, new supercomputers are more and more often designed to share performance among both CPUs and GPUs thus raising a problem of heterogeneous codes development.

We present Picador [14] — a tool for three-dimensional electromagnetic relativistic plasma simulation based on the particle-in-cell method, aimed at research of high intensity laser applications including generation of radiation with tailored

[*] Corresponding author.
E-mail address: meerov@vmk.unn.ru (I. Meyerov).

1877-7503/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.jocs.2012.08.012

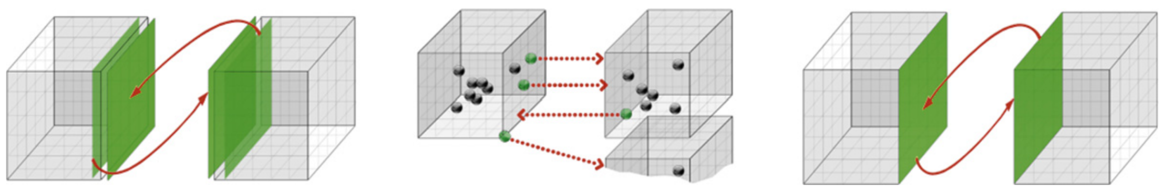

Fig. 1. The organization of the data exchanges. On the left — copy exchange of the magnetic field, on the middle — exchange of particles, on the right — add exchange of currents.

properties and laser-driven particle acceleration. The ultimate goal of our research is to create a high-performance code simultaneously utilizing all computational resources of heterogeneous cluster systems: CPUs, GPUs, accelerators such as Intel MIC. This paper addresses details of the PIC algorithm implementation for CPUs and GPUs and presents the up-to-date results concerning Picador efficiency and scalability on CPU- and GPU-enabled cluster systems. We describe the particle-in-cell algorithm in Section 2 and provide implementation details in Section 3. Results of computational experiments are presented in Section 4 and discussed in Section 5.

2. Methods

The particle-in-cell method is based on a self-consistent mathematical model representing plasma as an ensemble of negatively charged electrons and positively charged ions. The charged particles move under the influence of the electromagnetic field according to the relativistic equations of motion (all equations given in CGS units):

$$\frac{\partial \vec{p}}{\partial t} = q\left(\vec{E} + \frac{1}{c}\,\vec{v}\times\vec{B}\right) \qquad (1)$$

$$\frac{\partial \vec{r}}{\partial t} = \vec{v} = \frac{\vec{p}}{m}\left(1+\left(\frac{\vec{p}}{mc}\right)^2\right)^{-\frac{1}{2}}, \qquad (2)$$

where $m, q, \vec{r}, \vec{p}$, and $\vec{v}$ are the mass, charge, coordinates, momentum and velocity of the particle, respectively; $\vec{E}$, $\vec{B}$ are the electric and magnetic field values in the particle's position; c is the speed of light. The evolution of the electromagnetic field ($\vec{E}$, $\vec{B}$) is governed by Maxwell's equations:

$$\nabla\times\vec{B} = \frac{4\pi}{c}\vec{j} + \frac{1}{c}\frac{\partial \vec{E}}{\partial t} \qquad (3)$$

$$\nabla\times\vec{E} = -\frac{1}{c}\frac{\partial \vec{B}}{\partial t}, \qquad (4)$$

where $\vec{j}$ is the current density created by the particles.

The method assumes that the entire particle ensemble is represented by a smaller amount of so-called super-particles that have the same charge-to-mass ratio as the real particles. Since the charge and mass of the particle are only present in the equations of motion (1) and (2) as part of the ratio, this representation provides the equivalent plasma dynamics given the number of super-particles is large enough.

The simulation area is covered by a uniform space grid. The electric field $\vec{E}$ and current density $\vec{j}$ are represented by values in the nodes of the grid, while the magnetic field $\vec{B}$ is represented by values in the centers of the cells of the grid. The data for the particle ensemble is stored as well, each particle of a particular type is characterized by its position and velocity. The mass and charge are determined from the particle's type.

Each iteration of the computational loop corresponds to one time step and comprises four stages: weighting currents, integrating field equations, interpolating fields, integrating particle motion equations. The weighting currents stage implies computing of the current density based on the known positions and velocities of the charged particles. Each particle contributes to the current density values of only eight nearest grid nodes; the final values are determined by summarizing contributions of all particles. On the next stage the electric and magnetic fields on the grid are updated through numerical integration of Maxwell's equations with previously computed currents using the finite-difference time-domain (FDTD) method [15]. Next, linear interpolation of the electric and magnetic field grid values to the position of each particle is performed; for each particle only values corresponding to the eight nearest grid nodes are used. The resulting field values are used on the last stage for integrating the particle motion equations with the Boris method [1].

3. Implementation

Picador is written in C and C++in a fairly portable way, it supports building under Windows and Linux (and potentially under other POSIX-compliant systems). It is designed to be extensible; in particular, it allows to plug in different implementations of the main computational stages as well as message passing mechanisms. Simulation tasks are defined in the Tcl scripting language.

The implementation runs on cluster systems with one or several CPUs and GPUs on each node. The following execution scheme is used:

1. The problem is distributed among the nodes through MPI.
2. One or several MPI processes are launched on each node.
3. Each process uses either one or several CPU cores through OpenMP or a GPU through OpenCL.

3.1. Problem decomposition

The problem is decomposed according to a territorial principle: the simulation area is broken up into subareas (domains) handled in parallel by separate MPI processes. The process handling a particular domain keeps track of the data referring to the electromagnetic fields and particles that are located in the corresponding part of the physical space. Data corresponding to the boundaries between two or more domains is kept in all processes handling such domains. In addition, each process contain data referring to magnetic fields in the centers of cells bordering the handled domain.

To keep all domain information relevant, the data referring to the currents, fields and particles is exchanged. The exchanges are organized as follows (see Fig. 1). During current and field exchanges, each domain interacts with 6 neighbors. Each domain transfers values of the magnetic field corresponding to half-cell near boundary internal layer and receives values corresponding to half-cell near boundary outer layer from the domain having these values as internal. Values of current density correponding to the same positions are added. Particle exchange involves interaction with 26 neighbors, taking into account that when a particle leaves a domain it ends up in one of the 26 neighboring domains (due