Computer Physics Communications 185 (2014) 1364–1369

Contents lists available at ScienceDirect

Computer Physics Communications

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

Molecular dynamics for long-range interacting systems on graphic processing units

Tarcísio M. Rocha Filho *

Instituto de Física and International Center for Condensed Matter Physics, Universidade de Brasília, CP: 04455, 70919-970 - Brasília, Brazil

ARTICLE INFO

Article history:
Received 3 December 2012
Received in revised form
20 November 2013
Accepted 17 January 2014
Available online 24 January 2014

Keywords:
Molecular dynamics
Symplectic integrator
Long-range interaction

ABSTRACT

We present implementations of the numerical integration of systems with long-range interactions on graphic processing units for three N-body models with long-range interactions of general interest: the Hamiltonian Mean Field, Ring and two-dimensional self-gravitating models. We discuss the algorithms, speedups and errors using one and two GPU units. Speedups can be as high as 140 compared to a serial code, and the overall relative error in the total energy is of the same order of magnitude as for the CPU code. The number of particles used in the tests range from 10,000 to 50,000,000 depending on the model.

1. Introduction

The dynamics of systems with long-range interactions have been intensively studied over the last two decades due to their unusual and intriguing phenomenology, such as the existence of quasi-stationary non-Gaussian states with diverging life times with the number of particles, negative microcanonical heat capacity, inequivalence of ensembles and non-ergodicity [1–7]. Self-gravitating systems [8], non-neutral plasmas [9,10] and models as the Ring model [11,12] Hamiltonian Mean Field (HMF) [13], one-dimensional gravity (infinite sheets with uniform density) [14–16], two-dimensional gravity (infinite uniform rods) [17], Free Electron Laser [18] and plasma single wave models [19] are among many examples of systems with long-range forces. A pair potential interaction has a long-range if it scales for large distances as $r^{-\alpha}$ with $\alpha < d$, r the interparticle distance and d the spatial dimension. This slow decaying interparticle potential is responsible for the coupling of distant components of the system in such a way that all particles contribute to the dynamics of a given particle. For out of equilibrium situations, many of these studies rely on Molecular Dynamics (MD) simulations, i. e. solving numerically the Hamiltonian equations of motion for the N-particle system.

MD simulations have been extensively used to study many properties of these systems from first principles. Some numerical parallel algorithms were implemented in the literature with applications ranging from condensed matter to astrophysics [20–25]. The main computational effort relies on the computation of forces among N particles of the system, which usually scales as N^2. For large N the simulation time can become too large so some different approaches were introduced to simplify force calculations as the Particle-in-Cell [26], Particle–Mesh [27], hierarchical trees [28] and particle–mesh [29] methods, which were also ported to a GPU framework [30–37]. Another approach consists in s solving the Vlasov equation which describes the statistical properties for the system in the limit of a large number of particles [38,39], although finite N effects are lost in this case.

All particle force calculation are important for studying phenomena where first principles implementations are necessary to avoid spurious effects coming from different approximations, and have also been implemented using GPU computing [40–42]. A review of all algorithms implemented on GPU for astronomical systems is given in Ref. [43]. Ab initio MD codes with full particle force evaluation have been central in many important advances in the statistical mechanics of systems with long-range interactions (see [1] and references therein), and their implementation in the CUDA architecture [44] represented a leap forward as it allowed much large simulation times and particle numbers with many interesting results for the field [45–49]. Here we present a CUDA implementation of the MD algorithms used in the latter works, on both one and two GPUs to solve the corresponding Hamiltonian equations of motion for models in the field of Statistical Mechanics of systems with long-range interactions. The algorithms used can be efficiently extended for any number of GPUs. The simulations are performed without introducing any simplifying hypothesis with computational effort scaling as N^2. For some models the

* Tel.: +55 61 9215 8168; fax: +55 61 3307 2363.
E-mail addresses: marciano@fis.unb.br, marciano@deimos.fis.unb.br.

0010-4655/$ – see front matter © 2014 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.cpc.2014.01.008

explicit form of the pair interaction potential allows further simplifications with important reduction in computational time. We take advantage of the peculiarities of each model implemented to exploit the GPUs capabilities. The models here considered are all widely studied in the literature on long-range interacting systems.

The structure of the paper is the following: in Section 2 we describe the model systems studied in this paper. Section 3 presents the main algorithms and how they are implemented in CUDA, and in Section 4 we present the timings and speedups for the three models taken as examples, and discuss the relative error in the energy for each model. We close the paper with some concluding remarks in Section 5. In all these models the Kac factor $1/N$ ensures that the energy is extensive (although not additive) [1].

2. Models with long-range interactions

Here we present some simplified models that are discussed in the present work. The Ring model is composed of N particles with unit mass on a ring of radius R and interacting though a regularized gravitational potential with Hamiltonian [11,12]:

$$H = \frac{1}{2}\sum_{i=1}^{N} p_i^2 - \frac{1}{N}\sum_{i<j=1}^{N} \frac{1}{\sqrt{2}\sqrt{1-\cos(\theta_i-\theta_j)+\epsilon}}, \tag{1}$$

with ϵ a softening parameter introduced to avoid the divergence at zero distance and θ_i denotes the position angle of a particle on the circle. By considering the limit for large values of the parameter ϵ we obtain the Hamiltonian Mean Field (HMF) model with Hamiltonian [13]:

$$H = \frac{1}{2}\sum_{i=1}^{N} p_i^2 + \frac{1}{N}\sum_{i<j=1}^{N}\left[1-\cos\left(\theta_i-\theta_j\right)\right]. \tag{2}$$

The two-dimensional self-gravitating particles is composed of identical particles with unit mass. By solving the Poisson equation in two dimensions we obtain the Hamiltonian [17]:

$$H = \frac{1}{2}\sum_{i=1}^{N}(p_{x,i}^2 + p_{y,i}^2) + \frac{1}{N}\sum_{i<j=1}^{N}\log\left[(x_i-x_j)^2+(y_i-y_j)^2+\epsilon\right], \tag{3}$$

here $p_{x,i}, p_{y,i}, x_i$ and y_i are the x and y components of the momentum and the coordinates for the ith particle, respectively, and ϵ is again a (small) parameter used to avoid the divergence of the potential at zero distance.

3. Algorithms and CUDA implementation

The standard integration method for the numerical solution of Hamilton equations for the model systems described in the previous section is a fourth-order symplectic integrator [50]. Each time step requires three force calculations, being the most demanding step. All computations are fully performed on the GPUs. Once the force F_i on each particle is determined, the integration steps consist of a succession of velocity $v_i \to v + \xi\,\Delta t F_i$ and position $r_i \to r_i + \chi\,\Delta t\, v_i$ increments, where v_i and r_i stand for the velocities and position components of particle i, respectively, and ξ and χ are given constants specified by the integrator [50]. These steps are trivially parallelized, so we turn to present the specific CUDA implementation to compute the force for each case model taking advantage of its explicit form.

3.1. HMF model

From the Hamiltonian in Eq. (2) the potential energy and the force on particle i can be written respectively as

$$V = \frac{N}{2}\left[1 - M_x^2 - M_y^2\right], \tag{4}$$

and

$$f_i = \cos(\theta_i)M_y - \sin(\theta_i)M_x, \tag{5}$$

where the components of the "magnetization" are given by

$$M_x = \frac{1}{N}\sum_{i=1}^{N}\cos(\theta_i), \qquad M_y = \frac{1}{N}\sum_{i=1}^{N}\sin(\theta_i). \tag{6}$$

We note that due to the form of the interaction potential the symplectic integration time for the HMF scales with N in contrast to the usual N^2 scaling for the other two models. The most demanding part of an integration step is the computation of the sine and cosine of the position angles for each particle. For the force in Eq. (5) each term $\cos(\theta_i)$ and $\sin(\theta_i)$ is used twice: to compute M_x and M_y and to obtain the force on each particle from expression (5). To avoid redundant computations their values are first computed and stored in an array in the GPU global memory taking care to ensure coalesced memory access, which is an important issue in any CUDA implementation. A CUDA kernel is composed of blocks each with a given number of threads. Each thread in a block computes the value of the cosine and sine of a given particle and the corresponding values are also stored in shared memory. A reduction procedure is then used to compute the values of M_x and M_y and the force is obtained using the precomputed values of $\sin(\theta_i)$ and $\cos(\theta_i)$. The potential energy is trivially obtained from Eq. (4) and the kinetic energy is efficiently computed using a straightforward reduction procedure.

For two GPUs half of particle data (position, momenta and force) is stored in each GPU which then computes the magnetization components for its corresponding number of particles, and their sum give the total magnetization components. The forces on each particle are the trivially computed for the particles on each GPU, and the subsequent evolution is also performed independently by each GPU for its set of particles.

3.2. Ring model and self-gravitating 2D system

The computational time for symplectic integration of the Ring model scales as N^2. Here we follow a strategy similar to the one described in Ref. [42] based on a decomposition of the force calculation in tiles as depicted in Fig. 1. The force on particle i is obtained from Hamiltonian (1) as:

$$f_i = \sum_j f_{ij}, \qquad f_{ij} = \frac{1}{2\sqrt{2}N}\frac{\sin(\theta_i-\theta_j)}{\left(1-\cos(\theta_i-\theta_j)+\epsilon\right)^{3/2}}. \tag{7}$$

This last expression can be rewritten as

$$f_{ij} = \frac{1}{2\sqrt{2}N}\frac{\sin\theta_i\cos\theta_j - \cos\theta_i\sin\theta_j}{\left(1-\cos\theta_i\cos\theta_j-\sin\theta_i\sin\theta_j+\epsilon\right)^{3/2}}. \tag{8}$$

Each $\sin\theta_i$ and $\cos\theta_i$ for $i = 0,\ldots,N$ is computed and stored in global memory in order to avoid computing twice their values. Each tile has N_{threads} (the number of threads in a block) particles in the horizontal direction (index i in Fig. 1) and N_{threads} in the vertical direction (index j). The total number of tiles in each direction is thus $N_{tiles} = N/N_{\text{threads}}$. The algorithm can be expressed as:

1. Store $f_k = 0$ and the values of $\sin\theta_k$ and $\cos\theta_k$, $k = lN_{\text{threads}}, \ldots, (l+1)N_{\text{threads}} - 1$ in shared memory, with l the block number, and synchronize threads in the block.