Computational Materials Science 100 (2015) 173–180

Contents lists available at ScienceDirect

Computational Materials Science

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

Accelerating dissipative particle dynamics simulations for soft matter systems

Trung Dac Nguyen [a,*], Steven J. Plimpton [b]

[a] National Center for Computational Sciences, Oak Ridge National Laboratory, TN 37831, United States
[b] Sandia National Laboratories, Albuquerque, NM 87185, United States

ARTICLE INFO

Article history:
Received 22 September 2014
Received in revised form 30 October 2014
Accepted 31 October 2014
Available online 26 November 2014

Keywords:
Dissipative particle dynamics
LAMMPS
GPU acceleration
Hybrid CPU/GPU
Hybrid MPI/GPU
High-performance computing

ABSTRACT

Dissipative particle dynamics (DPD) is a coarse-grained particle-based simulation method that offers microscopic-scale insights into soft matter systems. We present an efficient implementation of a DPD model for graphical processing units (GPUs). As implemented in the LAMMPS molecular dynamics package, it can run effectively on current-generation supercomputers which often have hybrid nodes containing multi-core CPUs and (one or more) GPUs. Using efficient communication of information between the CPUs and GPUs, DPD interactions can be computed on the GPU while other portions of a full simulation model (boundary conditions, constraints, bonded interactions, diagnostic calculations, etc.) can be performed on the CPU. Our GPU-enhanced runs show a speedup of up to 9.5× versus many-core CPU simulations, and can run scalably across thousands of compute nodes. We briefly discuss how the new GPU implementation was validated against the CPU version for thermodynamics, diffusion, and hydrodynamic behavior. We also highlight large-scale models which the faster DPD implementation has enabled, for studies of monolayer self-assembly and thin-film instabilities.

1. Introduction

Dissipative particle dynamics (DPD) was first proposed by Hoogerbrugge and Koelman in 1992 as a particle-based method for studying hydrodynamic phenomena [1]. The statistical mechanics foundation of DPD was developed by Español and Warren in 1995 [2], followed by a rigorous analysis on the nature of its simulation parameters by Groot and Warren [3]. DPD has been widely used for studies of soft condensed matter systems, where the length scale of interest is greater than the atomistic scale but smaller than the characteristic structural dimensions, *e.g.* the scale of network connections or the spacings between assembled layers [3]. Owing to its inherent coarse-grained soft sphere model and its capability for capturing hydrodynamic interactions, DPD has also been used extensively in coupling particle-based simulation with continuum analyses, i.e. for multi-scale modeling. Examples include the study of fluid flows in microchannels [4–6], thin film evolution [7,8], self-assembly in surfactants and amphiphilic systems [9–15], structural and rheological properties of polymers [16–18] deformation of red blood cells [19] and membranes [20,21]. We refer the readers to reference [22] for a thorough review of the applications of DPD simulations to soft matter systems.

DPD is a relatively "cheap" computational model for two reasons. Its interactions are pairwise within a relatively short cut-off distance. And the pairwise interactions have a "soft" core which allows use of a larger timestep than for other potentials such as Lennard–Jones with its $1/r^{12}$ core. As highlighted below, these features have made it an attractive model for GPU implementation by several groups; pairwise potentials are more straight-forward to parallelize for a GPU than many-body potentials [23,24]. However, its soft core and the kinds of soft materials DPD is used to model, make it less amenable to the kinds of algorithmic acceleration techniques discussed elsewhere in this special issue. Methods such as parallel replica dynamics [25], temperature-accelerated dynamics [26], and kinetic Monte Carlo [27], enable dramatically longer timescale simulations by exploiting the rareness of discrete "events", defined as the crossing of well-defined energy barriers. These methods have thus been almost exclusively applied to solid-state materials. Soft materials, modeled by soft potentials, under non-equilibrium conditions, do not typically have easily-identifiable barriers and events. In this paper we therefore address the theme of the special issue by relying less on algorithmic insight and more on exploiting new hardware (GPUs) to achieve accelerated simulation capabilities. We do this for a class of coarse-grained models which have wide application in industrial

* Corresponding author.
E-mail address: ndactrung@gmail.com (T.D. Nguyen).

http://dx.doi.org/10.1016/j.commatsci.2014.10.068
0927-0256/

processing and more generally for simulation of liquids and soft matter at the mesoscale.

As is now well appreciated, the introduction of general-purpose graphics processing units (GPGPUs) to scientific computing (circa 2007) opened new opportunities for accelerating molecular dynamics (MD) codes. Exploiting the fine-grained parallelism offered by GPUs requires new MD methodologies. One trade-off is whether to offload only computationally intensive tasks to the GPU versus perform the entire simulation on the GPU. The former strategy (employed in this paper) is often pursued by legacy codes (*e.g.*, NAMD [28], GROMACS [29] and LAMMPS [30]), and the latter by newer codes (*e.g.*, HOOMD-Blue [31]).

Several successful efforts to accelerate DPD using GPUs have recently been described in the literature. To our knowledge, the first implementation of DPD to take full advantage of the single-instruction-multiple-threads (SIMT) environment of the GPU was the work of Phillips et al. [32] for HOOMD-Blue. In that work, a hash-function-based pseudo random number generator (PRNG) was used, previously developed by Steve Worley's group, to create a micro-stream of random numbers (RNs) per thread per kernel call. Their approach addressed issues with (1) memory storage for matching random numbers for each pair, (2) broadcasting the states of the PRNG among threads, (3) expensive arithmetic operations required for generating RNs, and (4) the low quality of built-in linear congruential RNGs for large systems and long simulations. Specific details on the PRNG were subsequently described in the work of Afshar et al. [33]. These ideas have been extended for accelerating DPD in codes using multiple GPUs [34], using message-passing-interface (MPI) parallelism [33], and using hybrid MPI/CUDA parallelism [14]. Notably, the GPU-accelerated version of a generalized DPD model recently described by Tang and Karniadakis [14] was shown to obtain a 10–30× speedup compared to a corresponding CPU-only version in LAMMPS [14]. This implementation was used in a large-scale mesoscopic study of amphiphilic systems in bulk and under soft confinement [15].

All of these DPD GPU implementations achieve their advertised speed-up when the entire MD simulation runs continuously on the GPU(s). By contrast, there is a package in LAMMPS (called GPU), which is designed to work on hybrid platforms, i.e. ones with compute nodes containing both CPUs and GPUs [35]. It allows pairwise potentials (and one many-body potential, Stillinger–Weber) as well as long-range electrostatic interactions to be computed on the GPU, concurrently with other bonded interactions (bonds, angles, dihedrals, etc.) computed on the CPU.

More importantly, the GPU package still achieves good performance while communicating updated atom properties (positions, forces) back-and-forth between the CPU(s) s and GPU each timestep. This is because the package allows multiple CPU cores to host a single GPU, which enables a greater degree of MPI-based parallelism across the multi-core CPUs of a large machine for these kinds of simulations. As a result, GPU utilization can be maximized and data transfers from one MPI task can be overlapped with GPU computations for other MPI tasks [23]. We note that complex simulations often involve significant computations that occur each timestep or every few timesteps, which at least in LAMMPS, have not been (and may never be) implemented for the GPU. Examples include specialized boundary conditions, application of constraint forces, simulation box deformations, calculation of diagnostic properties, etc. Using the GPU package, these computations still run efficiently on the CPU, using conventional MPI-based parallelization.

Many potentials have already been implemented for the GPU package in LAMMPS, including Coulombic and dipole–dipole interactions, Gay-Berne for ellipsoidal interactions, and the EAM, DLVO, and Stillinger–Weber potentials [36,37,23]. In this paper, we describe how the DPD model was implemented in the GPU package framework which required some unique issues to be addressed. The next sections give brief details of the implementation and illustrate its use in large-scale modeling of soft materials. The capabilities described here have been part of the general LAMMPS open-source release since April 2014.

2. Implementation

2.1. Equation of motion in DPD

In the standard DPD method, the equation of motion for particle i is given as [1–3]:

$$m_i \frac{d^2\mathbf{x}_i}{dt^2} = \sum_{j \neq i} \mathbf{F}_{ij}^C + \mathbf{F}_{ij}^D + \mathbf{F}_{ij}^R, \tag{1}$$

where m_i and $\mathbf{x}_i$ are the mass and position of particle i, respectively. $\mathbf{F}_{ij}^C$, $\mathbf{F}_{ij}^D$ and $\mathbf{F}_{ij}^R$ are the conservative force, drag force and random force, respectively, between particle i and its neighbor j within the cutoff distance r_c. These forces are defined as:

$$\mathbf{F}_{ij}^C = a_{ij} w_C(r_{ij}) \mathbf{e}_{ij} \tag{2}$$

$$\mathbf{F}_{ij}^D = -\gamma w_D(r_{ij})(\mathbf{e}_{ij} \cdot \mathbf{v}_{ij})\mathbf{e}_{ij} \tag{3}$$

$$\mathbf{F}_{ij}^R = \sigma w_R(r_{ij}) \xi_{ij} \Delta t^{-1/2} \mathbf{e}_{ij} \tag{4}$$

where $\mathbf{r}_{ij} = r_{ij}\mathbf{e}_{ij}$ is the distance vector between particles i and j, a_{ij} is the interaction strength, $\mathbf{v}_{ij} = \mathbf{v}_i - \mathbf{v}_j$, Δt is the integration time step, and ξ_{ij} is a random variable with Gaussian characteristics: $\langle \xi_{ij}(t) \rangle = 0$ and $\langle \xi_{ij}(t)\xi_{kl}(t') \rangle = (\delta_{ik}\delta_{jl} + \delta_{il}\delta_{jk})\delta(t - t')$ [3]. Importantly, the random variable is symmetric for every pair $ij : \xi_{ij}(t) = \xi_{ji}(t)$. $w_C(r_{ij})$, $w_D(r_{ij})$ and $w_R(r_{ij})$ are the weighing functions designed to vanish for $r \geqslant r_c$. In Eq. (4), the random force magnitude is inversely proportional to the time step, $\Delta t^{-1/2}$, to take into account the time discretization of the numerical integrator [3].

For mesocopic coarse-grained models, for which DPD was developed, the conservative weighing function $w_C(r_{ij})$ is often chosen as, but not restricted to, a soft-core potential:

$$w_C(r_{ij}) = 1 - \frac{r_{ij}}{r_c}, \tag{5}$$

which allows particles to pass through each other. It is the soft-core model that allows for substantially bigger integration time steps compared to those used for atomic- and molecular-scale models. The random and drag weighing functions, and their corresponding amplitudes, are related to each other via the fluctuation–dissipation theorem [2,3]:

$$w_R^2(r_{ij}) = w_D(r_{ij}) \tag{6}$$

$$\sigma^2 = 2\gamma k_B T \tag{7}$$

In the original formulation, ξ_{ij} is a normally distributed random variable with zero mean and unit variance, i.e., $N(0, 1)$. However, it was shown that a uniformly distributed variable, $U(-0.5, 0.5)$, with amplitude multiplied by $\sqrt{12}$, yields indistinguishable results [3], an effect we examine in Section 3.2.

The relationship between the drag force and random force (Eqs. (6) and (7)) allows DPD to sample from the canonical ensemble [2]. Because the drag and random forces force between two interacting particles satisfy Newton's third law, they serve as a momentum-conserving thermostat. This is in contrast to Brownian Dynamics where the random forces applied to individual particles are uncorrelated. As a result, DPD is the current method of choice for simulating mesoscopic systems where hydrodynamic interactions are relevant.