Computer Physics Communications 218 (2017) 10–16

Contents lists available at ScienceDirect

Computer Physics Communications

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

Efficient molecular dynamics simulations with many-body potentials on graphics processing units

Zheyong Fan [a,b,*], Wei Chen [c,*], Ville Vierimaa [b], Ari Harju [b]

[a] School of Mathematics and Physics, Bohai University, Jinzhou, China
[b] COMP Centre of Excellence and Helsinki Institute of Physics, Department of Applied Physics, Aalto University, Helsinki, Finland
[c] Computer Network Information Center, Chinese Academy of Sciences, P.O. Box 349, 100190 Beijing, China

ARTICLE INFO

Article history:
Received 12 October 2016
Received in revised form 2 May 2017
Accepted 4 May 2017
Available online 10 May 2017

Keywords:
Molecular dynamics simulation
Many-body potential
Tersoff potential
Stillinger–Weber potential
Graphics processing units
Virial stress
Heat current

ABSTRACT

Graphics processing units have been extensively used to accelerate classical molecular dynamics simulations. However, there is much less progress on the acceleration of force evaluations for many-body potentials compared to pairwise ones. In the conventional force evaluation algorithm for many-body potentials, the force, virial stress, and heat current for a given atom are accumulated within different loops, which could result in write conflict between different threads in a CUDA kernel. In this work, we provide a new force evaluation algorithm, which is based on an explicit pairwise force expression for many-body potentials derived recently (Fan et al., 2015). In our algorithm, the force, virial stress, and heat current for a given atom can be accumulated within a single thread and is free of write conflicts. We discuss the formulations and algorithms and evaluate their performance. A new open-source code, GPUMD, is developed based on the proposed formulations. For the Tersoff many-body potential, the double precision performance of GPUMD using a Tesla K40 card is equivalent to that of the LAMMPS (Large-scale Atomic/Molecular Massively Parallel Simulator) molecular dynamics code running with about 100 CPU cores (Intel Xeon CPU X5670 @ 2.93 GHz).

1. Introduction

Molecular dynamics (MD) simulation is one of the most important numerical tools in investigating various physical properties of materials. Many applications using MD simulation demand high performance computing. In the past decade, the computational power of general-purpose graphics processing units (GPUs) has been exploited to accelerate many MD simulations. Not only existing MD codes and libraries, such as AMBER [1], Gromacs [2,3], LAMMPS [4–6], NAMD [7], and OpenMM [8] have been benefited from utilizing GPUs as accelerators, but also new codes, such as HOOMD-blue [9–11], HALMD [12], and RUMD [13], have been built from the ground up to achieve high performance using one or more GPUs.

Most of the previous relevant works have only considered pairwise potentials, or a special many-body potential, namely, the embedded atom method [14–16], which are relatively simple to implement on GPUs. GPU-acceleration of many-body potentials such as the Tersoff [17], Stillinger–Weber [18], and Brenner [19] potentials, which play an important role in modelling various materials, is more challenging and has only attracted some attention recently [20–25]. Taking three-body interaction as an example, a naive implementation of the force evaluation function, as usually done in a serial CPU code, requires accumulating the forces on three different atoms within a single thread. In a GPU kernel with many threads, each atom is usually associated with one thread and the force accumulation for an atom from the thread it belongs to will conflict with the force accumulation for the same atom from another thread. This causes a problem called write conflict where two threads try to write data simultaneously into the same global memory [26]. One way to avoid write conflict is to use atomic operations, which are usually quite slow and can also introduce randomness in the computation, which is undesirable for debugging.

There have been some proposals to avoid using atomic operations. Hou et al. [20] proposed an algorithm for implementing the Tersoff potential on a GPU, which has achieved impressive performance, but requires using a special fixed neighbour list and is thus not quite flexible. Brown and Yamada [21] proposed a flexible GPU-implementation of the Stillinger–Weber potential within LAMMPS, which is free of write conflicts. A similar proposal was given by Knizhnik et al. [22]. Recently, Höhnerbach et al. [23] developed a vectorization scheme to achieve performance portability across various parallel computing platforms for the Tersoff potential within LAMMPS. GPU-acceleration of the more complicated second-generation REBO potential [19] has also been studied

* Corresponding authors.
E-mail addresses: brucenju@gmail.com (Z. Fan), weichen@cnic.cn (W. Chen).

http://dx.doi.org/10.1016/j.cpc.2017.05.003
0010-4655/

by Tredak et al. [24]. In a recently published work [25] (after we submitted this paper), Nguyen reported significant speedups for MD simulations with Tersoff-type potentials using one or more high-end GPUs.

Here, we propose a general algorithm of force evaluation for many-body potentials and present details of its GPU-implementation and performance. The new force evaluation algorithm is based on an explicit pairwise force expression for many-body potentials derived recently [27]. In this approach, the force, virial stress, and heat current for a given atom are well defined and can be accumulated within a single thread. Therefore, write conflict is absent by construction. To be specific, we discuss the algorithm explicitly in terms of the Tersoff potential, but performance evaluation is made for both the Tersoff potential and the Stillinger–Weber potential. The implementation is done based on a previous work [28], and the resulting code, which we call GPUMD (Graphics Processing Units Molecular Dynamics), will be made public soon. Using silicon crystal as a test system, we measure the performance of GPUMD and compare it with LAMMPS.

2. Formulations and algorithms

2.1. The Tersoff many-body potential

Although the method to be introduced is applicable to any many-body potential, it is beneficial to start with an explicit example, which is taken as the widely used Tersoff potential. Generalizations to other many-body potentials will be discussed later.

The total potential energy for a system of N atoms described by the Tersoff potential can be written as [17]

$$U = \frac{1}{2}\sum_i \sum_{j\neq i} U_{ij}, \tag{1}$$

where

$$U_{ij} = f_C(r_{ij})\left(f_R(r_{ij}) - b_{ij} f_A(r_{ij})\right), \tag{2}$$

$$b_{ij} = \left(1 + \beta^n \zeta_{ij}^n\right)^{-\frac{1}{2n}}, \tag{3}$$

$$\zeta_{ij} = \sum_{k\neq i,j} f_C(r_{ik}) g_{ijk}, \tag{4}$$

$$g_{ijk} = 1 + \frac{c^2}{d^2} - \frac{c^2}{d^2 + (h - \cos\theta_{ijk})^2}. \tag{5}$$

Here, β, n, c, d, and h are material-specific parameters and θ_{ijk} is the angle formed by $\boldsymbol{r}_{ij}$ and $\boldsymbol{r}_{ik}$, which implies that

$$\cos\theta_{ijk} = \cos\theta_{ikj} = \frac{\boldsymbol{r}_{ij}\cdot\boldsymbol{r}_{ik}}{r_{ij} r_{ik}}. \tag{6}$$

Our convention is that $\boldsymbol{r}_{ij} \equiv \boldsymbol{r}_j - \boldsymbol{r}_i$ represents the position difference pointing from atom i to atom j. The magnitude of $\boldsymbol{r}_{ij}$ is denoted as r_{ij}.

As in many empirical potentials, the energy U_{ij} consists of a repulsive part $f_R(r_{ij})$ and an attractive part $-b_{ij} f_A(r_{ij})$. The many-body nature of the Tersoff potential is embodied in the bond order function b_{ij} appearing in the attractive part, the value of which depends not only on $\boldsymbol{r}_i$ and $\boldsymbol{r}_j$, but also on the positions of other atoms near atom i.

The function $f_C(r_{ij})$ is a cutoff function, which is only nonzero when r_{ij} is less than a cutoff distance. Therefore, a Verlet neighbour list can be used to speed up the force evaluation. For uniform cutoff, the standard cell list method is very efficient, although more sophisticated methods perform better for systems with large size disparities [29].

For simplicity, we have presented the original Tersoff potential formulation in a form suitable for single-element systems. Our algorithm and implementation are more general, which can treat systems with more than one type of atom or systems described by a modified formulation of the Tersoff potential.

2.2. The conventional method of implementing the tersoff potential

Due to the three-body nature of the Tersoff potential, the conventional method for evaluating the interatomic forces is significantly different from that in the case of a simple two-body potential. Algorithm 1 presents a pseudo code for the conventional method as implemented in most existing MD codes such as LAMMPS [4]. The following symbols are used:

- N: number of atoms
- U_i: potential energy of atom i
- $\boldsymbol{F}_i$: total force on atom i
- $\mathbf{W}_i$: per-atom virial stress of atom i
- NN_i: number of neighbour atoms of atom i
- NL_{im}: index of the mth neighbour atom of atom i
- $\boldsymbol{J}_i$: per-atom heat current of atom i.

Algorithm 1 Pseudo code for the conventional method of evaluating many-body force and related quantities.

1: **for** $i = 0$ to $N-1$ **do**
2: Initialize U_i, $\boldsymbol{F}_i$, and $\mathbf{W}_i$ to zero
3: **end for**
4: **for** $i = 0$ to $N-1$ **do**
5: **for** $m = 0$ to $\mathrm{NN}_i - 1$ **do**
6: $j \leftarrow \mathrm{NL}_{im}$
7: $U_i \leftarrow U_i + \frac{1}{2}U_{ij}$
8: $\boldsymbol{F}_i \leftarrow \boldsymbol{F}_i + \boldsymbol{F}_i^{(ij)}$
9: $\boldsymbol{F}_j \leftarrow \boldsymbol{F}_j + \boldsymbol{F}_j^{(ij)}$
10: $\mathbf{W}_i \leftarrow \mathbf{W}_i - \frac{1}{2}\boldsymbol{r}_{ij}\otimes\boldsymbol{F}_i^{(ij)}$
11: **for** $n = 0$ to $\mathrm{NN}_i - 1$ **do**
12: $k \leftarrow \mathrm{NL}_{in}$
13: **if** $k = j$ **then**
14: Continue
15: **end if**
16: $\boldsymbol{F}_i \leftarrow \boldsymbol{F}_i + \boldsymbol{F}_i^{(ijk)}$
17: $\boldsymbol{F}_j \leftarrow \boldsymbol{F}_j + \boldsymbol{F}_j^{(ijk)}$
18: $\boldsymbol{F}_k \leftarrow \boldsymbol{F}_k + \boldsymbol{F}_k^{(ijk)}$
19: $\mathbf{W}_i \leftarrow \mathbf{W}_i + \frac{1}{3}\left(\boldsymbol{r}_{ij}\otimes\boldsymbol{F}_j^{(ijk)} + \boldsymbol{r}_{ik}\otimes\boldsymbol{F}_k^{(ijk)}\right)$
20: $\mathbf{W}_j \leftarrow \mathbf{W}_j + \frac{1}{3}\left(\boldsymbol{r}_{ij}\otimes\boldsymbol{F}_j^{(ijk)} + \boldsymbol{r}_{ik}\otimes\boldsymbol{F}_k^{(ijk)}\right)$
21: $\mathbf{W}_k \leftarrow \mathbf{W}_k + \frac{1}{3}\left(\boldsymbol{r}_{ij}\otimes\boldsymbol{F}_j^{(ijk)} + \boldsymbol{r}_{ik}\otimes\boldsymbol{F}_k^{(ijk)}\right)$
22: **end for**
23: **end for**
24: **end for**
25: **for** $i = 0$ to $N-1$ **do**
26: $\boldsymbol{J}_i \leftarrow \mathbf{W}_i \cdot \boldsymbol{v}_i$
27: **end for**

In Algorithm 1, the potential energy $U_i \equiv \sum_{j\neq i} U_{ij}/2$ is accumulated in line 7, the two-body parts of the force and per-atom virial stress are accumulated in lines 8–9 and 10, respectively, and the many-body parts of the force and per-atom virial stress are accumulated in lines 16–18 and 19–21, respectively. Last, in line 26, the per-atom heat current is calculated from the per-atom virial stress and velocity.

The forces defined in the pseudo code can be explicitly written as

$$\boldsymbol{F}_i^{(ij)} = -\frac{1}{2}\frac{\partial}{\partial\boldsymbol{r}_i}\left(f_C(r_{ij}) f_R(r_{ij})\right) + \frac{1}{2} b_{ij}\frac{\partial}{\partial\boldsymbol{r}_i}\left(f_C(r_{ij}) f_A(r_{ij})\right), \tag{7}$$