Computer Physics Communications 182 (2011) 1974–1978

Contents lists available at ScienceDirect

Computer Physics Communications

www.elsevier.com/locate/cpc

Molecular dynamics simulations of the relaxation processes in the condensed matter on GPUs

I.V. Morozov [a,b,*], A.M. Kazennov [a,b], R.G. Bystryi [a,b], G.E. Norman [a,b], V.V. Pisarev [a,b], V.V. Stegailov [a,b]

[a] Joint Institute for High Temperatures of Russian Academy of Sciences, Izhorskaya 13, build. 2, Moscow, 125412, Russia
[b] Moscow Institute of Physics and Technology, Institutskii per., 9, Dolgoprudny 141700, Moscow Region, Russia

ARTICLE INFO

Article history:
Received 30 July 2010
Received in revised form 11 December 2010
Accepted 20 December 2010
Available online 23 December 2010

Keywords:
Molecular dynamics
GPU computing
Embedded atom method
Nonequilibrium process
Metastable state

ABSTRACT

We report on simulation technique and benchmarks for molecular dynamics simulations of the relaxation processes in solids and liquids using the graphics processing units (GPUs). The implementation of a many-body potential such as the embedded atom method (EAM) on GPU is discussed. The benchmarks obtained by LAMMPS and HOOMD packages for simple Lennard-Jones liquids and metals using EAM potentials are presented for both Intel CPUs and Nvidia GPUs. As an example the crystallization rate of the supercooled Al melt is computed.

1. Introduction

The method of molecular dynamics (MD) is widely used to study static and dynamic properties of the condensed matter [1,2]. In particular this method is indispensable for studying the relaxation of nonequilibrium and metastable states which play essential role in the impulse loading processes such as shock compression, laser ablation, etc. Examples of such simulations are the studies of the shock wave fracture in solids [1,3], void formation and growth in liquids under negative pressures [4], melting of solids [5,6]. These simulations are computationally intensive due to the large MD cell size (many particles), long relaxation period (many time steps) or large number of statistical averages. The general approach for studying the relaxation of metastable states is presented in [2]. Its basic idea is to consider a small system volume but average the results over an ensemble of initial states. Namely a bunch of MD runs should be performed starting from different microscopical configurations which correspond to the same (nonequilibrium) macroscopical state. Then either the statistical averaging is performed or the distribution of the random events (such as phase transitions) is built.

In the context of this approach the code can be parallelized at two levels: particle or domain decomposition for the computation of a single trajectory and parallel run of the bunch of trajectories at different cluster (Grid) nodes. The second method does not require interprocess communications and therefore has an ideal parallel efficiency. Moreover it is fail-safe with respect to a failure of a particular trajectory calculation. As to the hybrid computer systems based on GPUs, it is possible to use GPUs to speed up the calculation of a single trajectory at the first level. Then the whole bunch of trajectories can be run on a multi-GPU workstation or a hybrid cluster.

Recently essential progress was achieved in the area of general purpose GPU computations. Contemporary GPUs provide tremendous computational power but at the same time require specific highly parallelized codes. There are few MD packages that support GPU computing: LAMMPS [7], NAMD/VMD [8], HOOMD [9], MDGPU [10], GROMACS + OpenMM [11], etc. This area is developing rapidly although at present only simple interaction potentials are implemented on GPUs.

In this paper we consider LAMMPS and HOOMD codes as they are mostly suitable for the condensed matter physics. LAMMPS has a long history and it was successfully used for similar problems of the nonequilibrium condensed matter simulations (see e.g. [1,2]). It provides good parallelizing efficiency especially at large computing clusters. The GPU-based force fields are supported in LAMMPS since August 2009 but only the Lennard-Jones (LJ) and few other simple potentials are implemented.

HOOMD is a relatively new project which was designed for GPU from the beginning. It is based on Nvidia CUDA [12] environment being highly optimized with respect to the specific Nvidia GPU architecture. To reduce the host-device data transfer most MD algorithms such as building of the neighbor list, calculation

* Corresponding author at: Joint Institute for High Temperatures of Russian Academy of Sciences, Izhorskaya 13, build. 2, Moscow, 125412, Russia.
E-mail address: morozov@ihed.ras.ru (I.V. Morozov).

0010-4655/$ – see front matter
doi:10.1016/j.cpc.2010.12.026

of interparticle forces and solution of the equation of motion are performed on GPU.

In the second section we discuss the benchmarks obtained with LAMMPS and HOOMD on both CPU and GPU for the equilibrium LJ liquid. As the condensed matter simulations considered below require more realistic interaction models then simple pair potentials, the third section is concerned with our own implementation of the Embedded Atom Method (EAM) [13] on GPU in the framework of the HOOMD package. Finally the benchmarks for the EAM code and the results for the Al crystallization rate are presented.

2. Hardware and software and used for benchmarks

The following computational systems were used for benchmarks.

1. *JIHT Hybrid Workstation*:
   CPU: Intel Core 2 Quad Q6600, 2.4 GHz; GPUs: 2 × Nvidia GeForce 260GTX, 1 Gb RAM, 192 cores; RAM: 4 Gb DDR2; OS: OpenSuse 10.3.
2. *MIPT Hybrid Workstation*:
   CPUs: 2 × Intel Xeon E5520 (Nehalem), 2.27 GHz, 8 Mb L3; GPUs: 2 × Nvidia GeForce 480GTX 1.5 Gb RAM, 480 cores, 2 × Nvidia Tesla C2050, 2 × Nvidia Tesla C1060; RAM: 32 Gb DDR3; OS: Ubuntu 10.
3. *MIPT Conventional Cluster*:
   CPUs: 268 × Intel Xeon 5160 (Woodcrest), 3.0 GHz, 4 Mb L2; RAM: 134 × 4 Gb DDR2; Interconnect: Myrinet; OS: CentOs 5.

We used HOOMD ver. 0.8.1 compiled with GCC 4.1.2 and LAMMPS (14 Oct. 2009) compiled with Intel 11.0 compiler for benchmarks on JIHT Hybrid Workstation; HOOMD ver. 0.9.0 compiled with GCC 4.3.4 and LAMMPS (20 May 2010) compiled with Intel 11.0 compiler on MIPT Hybrid Workstation; LAMMPS (14 Oct. 2009) compiled with Intel 9.1 compiler on MIPT Conventional cluster. Nvidia CUDA architecture 1.3 was used for all GPU applications.

3. Benchmarks for the Lennard-Jones potential

Let us consider a LJ liquid in the MD box with periodic boundary conditions. Using the LJ units the pairwise interaction potential can be written as $U(r) = 4(r^{-12} - r^{-6})$. The Berendsen thermostat is used to achieve the equilibrium but it is switched off for the main MD run. The average temperature and density are $T = 1.0$ and $\rho = 0.19$, the time step is $\Delta t = 10^{-4}$, the cutoff parameter is $r_{\text{cut}} = 3$. The equations of motion was integrated using the Velocity-Verlet algorithm. The skin width of $r_{\text{skin}} = 0.8$ beyond the cutoff is taken for the neighbor list construction which provides the best performance.

The time needed to perform a single MD step (the lower the better) is presented in Fig. 1 depending on the number of particles N_{part} for the constant density. The number of CUDA threads is equal to the number of particles as each thread calculates the force on a particular particle. It is seen that for a small number of particles ($N_{\text{part}} < 500$) the GPU is underloaded and provides worse performance than CPU. At higher particle number the GPU takes over and the performance ratio between 260GTX GPU and Q6600 CPU (using 4 cores) reaches the value of 30. It should be noted however that LAMMPS uses double floating point precision for CPU version whereas HOOMD uses the single one for both CPU and GPU.

When we switched from the old JIHT Hybrid Workstation to the newer MIPT Hybrid Workstation, the maximal GPU-CPU ratio of 7.6 was found (Fig. 2) for Nvidia GeForce 480GTX GPU and two Intel Xeon E5520 CPUs. For CPU computations we ran 16 LAMMPS MPI processes on 8 cores using hyperthreading which gives the best overall performance. The processes communicated via the shared memory. At high particle numbers the communication delays are negligible.

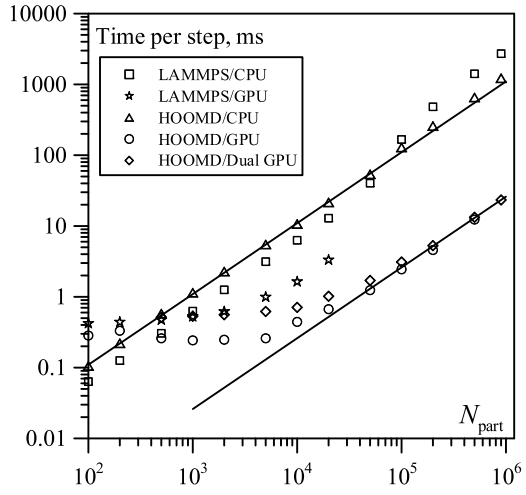

Fig. 1. The time required for a single MD step depending on the number of particles in the LJ liquid. The results are obtained on JIHT Hybrid Workstation running LAMMPS on 1 core of Q6600 CPU (squares), LAMMPS on 260GTX GPU (stars), HOOMD on 1 core of Q6600 CPU (triangles), HOOMD on 260GTX GPU (circles) and HOOMD on 2 × 260GTX GPUs (diamonds). Lines represent linear fits.

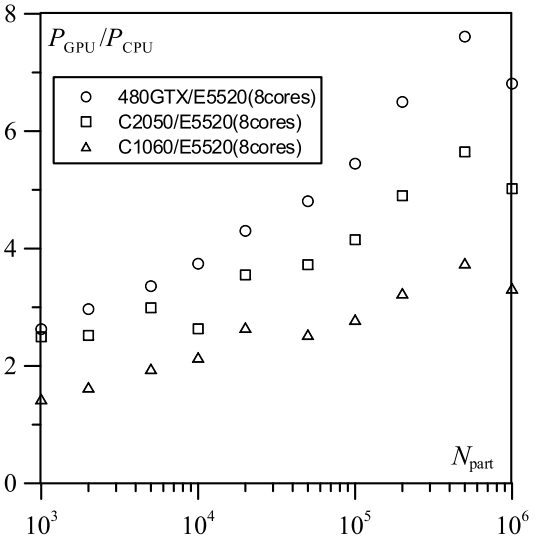

Fig. 2. The ratio between GPUs and CPU performances depending on the number of particles in the LJ liquid. The results are obtained on the MIPT Hybrid Workstation. The performances of HOOMD running on 480GTX GPU (circles), C2050 GPU (squares) and C1060 GPU (triangles) are related to the performance of LAMMPS running on 8 cores of E5520 CPU.

The maximal number of particles is limited by the amount of GPU memory needed to store the neighbor lists. For the given parameters it is approximately 10^6 particles per 1 GiB GPU memory (HOOMD). For unknown reasons LAMMPS imposes more strict constraints of about 20 000 particles on 1 GiB 260GTX GPU. Moreover for all N_{part} values LAMMPS shows worse performance (Fig. 1). Unfortunately we failed to run LAMMPS on the newer GPUs.

As seen from Fig. 1 the work distribution between two GPUs enabled in HOOMD does not give any performance gain possibly due to the large amount of inter-GPU communications.

4. Implementation of the embedded atom method on GPU

The EAM is a semi-empirical many-body potential which provides appropriate description for the atomic interactions in solids. Within this approach the interaction between each atom pair depends not only on the interparticle distance but also on the induced electron density. The general from of the potential reads [13]

$$U = U^{\mathrm{em}} + U^{\mathrm{pair}} = \sum_i F_i\left(\rho_i^{\Sigma}\right) + \sum_i \sum_{i<j} \Phi_{ij}(r_{ij}), \tag{1}$$

where $\rho_i^{\Sigma} = \sum_{i\neq j} \rho_i(r_{ij})$ is the total electron density induced on ith atom by all others, r_{ij} is the distance between ith and jth atoms, F_i is the embedding function and Φ_{ij} is the pair potential.

Thus one needs $\rho(r)$ and $F(\rho)$ for each atom type and $\Phi(r)$ for each pair of atom types to define the potential. Typically these functions are tabulated (see e.g. [14,15]) so that the interpolation is needed to calculate the energy of a particular system state. In our implementation of EAM potential on Nvidia GPUs we take advantage of the linear hardware interpolation for this purpose. It is less accurate than the 4th order spline interpolation used in LAMMPS but (as will be seen from the results) provides acceptable accuracy for our problems.

The force on the ith particle depends on the derivatives of the corresponding functions which can be tabulated as well. The force related to the embedding energy U^{em} is given by

$$\mathbf{f}_i^{\mathrm{em}} = -\frac{\partial U^{\mathrm{em}}}{\partial \mathbf{r}_i} = -\sum_{j\neq i} \frac{\mathbf{r}_{ij}}{r_{ij}}\left[F_i'\left(\rho_i^{\Sigma}\right)\rho_j'(r_{ij}) + F_j'\left(\rho_j^{\Sigma}\right)\rho_i'(r_{ij})\right]. \tag{2}$$

From the computational point of view the main difference between this equation and the corresponding equation for a pairwise interaction is that the value of $\mathbf{f}^{\mathrm{em}}$ should be computed in two stages. First the array of the electron densities ρ_i^{Σ} or the quantities $F_i'(\rho_i^{\Sigma})$ is calculated and stored in the (GPU) memory. Second the total force is obtained for each particle according to (2). It requires synchronization of all threads between the stages which can be done only by running two CUDA kernels sequentially. Moreover the memory operations are rather expensive on GPU.

Thus one can see that the many-body force field implementation on GPU turns out not to be a straightforward generalization of a pairwise potential.

5. Benchmarks for the embedded atom method

An equilibrium crystal state for Cu is prepared with the temperature $T = 300$ K for benchmarks. The results are presented in Fig. 3. The EAM tables are taken from [14]. The time step of $\Delta t = 1$ fs is used and the neighbor list skin width is set to $r_{\mathrm{skin}} = 0.3$ Å.

It is seen that in contrast to LJ potential the GPU is profitable even at the small particle number as the amount of computations at each time step is greater for EAM and the parallelizing overhead is relatively smaller.

The similar results are obtained for Al (Fig. 4) where the initial state is the supercooled liquid at the temperature $T = 750$ K and pressure $P = 50$ Kbar. The melting temperature for Al at this pressure is $T = 1180$ K. These parameters correspond to the initial state for the real physical problem discussed in the next section. The trajectory length used for benchmarks is small enough so that the crystallization does not occur.

The results for different metals are summarized in Fig. 5 where the GPU to CPU performance ratio is presented. One can see that the maximal ratio for 8 CPU cores is about 8 which is the same as for LJ potential. This means that despite of the problems with calculation of embedding forces discussed above the GPU implementation of the EAM potential is not less effective then LJ. It

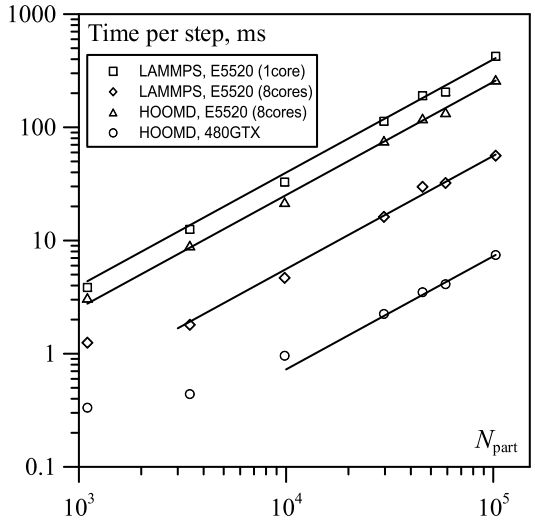

Fig. 3. Time per MD step depending on the number of particles. Solid Cu is simulated using EAM potential on MIPT Hybrid Workstation. LAMMPS runs on two E5520 CPUs using 1 core (squares) and 8 cores (16 MPI processes) (diamonds), HOOMD runs on two E5520 CPUs using 8 cores (triangles) and on 480GTX GPU (circles). Lines represent linear fits.

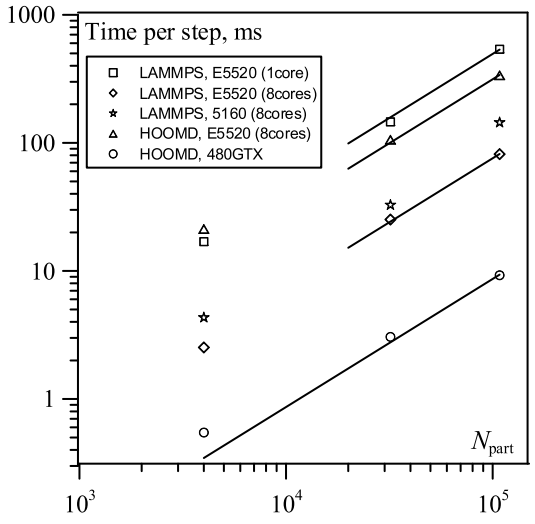

Fig. 4. Time per MD step depending on the number of particles. Supercooled liquid Al is simulated using EAM potential on MIPT Hybrid Workstation. The symbols have the same meaning as in Fig. 3 except the stars which related to the results obtained by LAMMPS on 8 cores (2 nodes) of MIPT Conventional Cluster.

should be noted that according to our tests, the EAM potential works 2–3 times slower than LJ on the old G200 generation of Nvidia GPUs. The essential increase of the EAM performance which makes it as fast as LJ on the present Fermi architecture can be related to the introduction of the L2 cache.

As the real computations are performed usually on supercomputer clusters we made a comparison of the performance of a single the Nvidia 480GTX GPU (using HOOMD) with different number of cluster cores (using LAMMPS). The interprocess communications on the cluster are provided by MPICH MPI ver. 1.2.7. Taking into account the MPI communication losses we found that a single GPU can compete with about 120 Xeon 5160 cores for $N_{\mathrm{part}} = 108\,000$ (Fig. 6).

6. Crystallization of the supercooled Al melt

In this section we consider the real problem of calculation of the crystallization rate of the supercooled Al melt. The supercooled

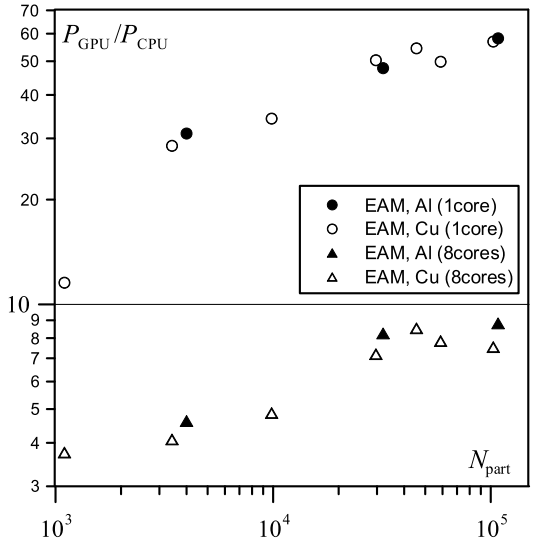

Fig. 5. GPU to CPU performance ratio obtained on MIPT Hybrid Workstation for Cu (open symbols) and Al (filled symbols). HOOMD running on 480GTX GPU is compared with LAMMPS running on 1 core (circles) and 8 cores (triangles) of two E5520 CPUs.

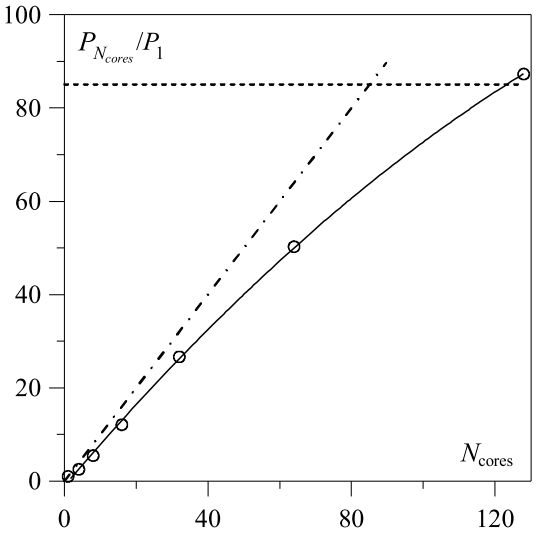

Fig. 6. The performance of LAMMPS running on N_{cores} nodes of MIPT Conventional Cluster related on the performance on a single node (circles and solid line fit). The dash-dotted line correspond to the ideal parallelizing efficiency. For comparison the corresponding performance of HOOMD running on a single 480GTX GPU of MPI Hybrid Workstation is given as the dashed line.

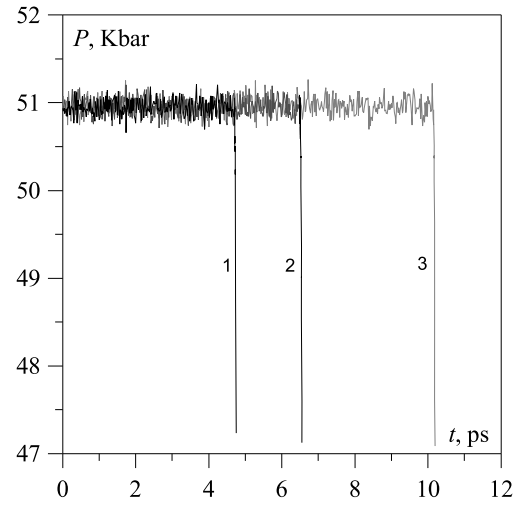

Fig. 7. Dependence of the pressure on time for three trajectories for slightly different microscopical states. The pressure drops 1–3 correspond to the crystallization events.

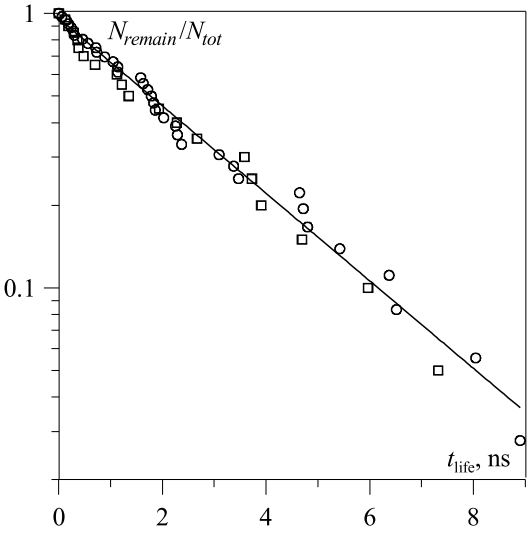

Fig. 8. The number of trajectories on which the metastable state lives longer than t_{life} related to the total number of trajectories: squares are the results obtained by LAMMPS on CPU (MIPT Conventional Cluster), circles are obtained by HOOMD on GPUs (MIPT Hybrid Workstation and MIPT hybrid cluster). The strait line is the exponential fit.

metallic liquids appear for example after the crack formation induce by a shock wave. This problem was studied both theoretically and numerically (see e.g. [16]) but there are still open questions which require simulations. The results discussed in this section will allow us also to compare the precision of the CPU and GPU algorithms.

The simulations scheme is as follows. First we obtain the equilibrium Al liquid with the temperature T above the melting temperature T_m. Then using the thermostat we decrease the temperature to a certain value below T_m. For the crystallization process is slower then the thermal equilibration, such supercooled state can be easily prepared and saved for the next simulation stage.

As the crystallization in a single MD trajectory is a random event, one needs to calculate the bunch of trajectories which are different in the microscopical point of view but correspond to the same macroscopical state. There are few means to create such trajectories [2]. In the present study we use slightly different integration time steps for different trajectories starting from the same supercooled initial configuration obtained at the previous stage. The difference between the time steps is of the order of $10^{-3}\Delta t$ which does not affect the accuracy of each trajectory calculations but makes the trajectories statistically independent due to the Lyapunov instability, provided the dynamical memory time is smaller then the mean supercooled liquid lifetime. Thus for each trajectory in the bunch the time step is determined as $\Delta t_k = \Delta t(1 + 10^{-3}k)$ where k is the trajectory number and Δt is about one femtosecond.

The crystallization moments can be found by determination of the pressure drops (see Fig. 7). It gives us the supercooled liquid lifetime t_{life} for each trajectory. Then the distribution of the lifetimes is build. The result is shown in Fig. 8 where the ordinate axis corresponds to the number of trajectories N_{remain} where the crystallization does *not* occur till the time t_{life} related to the total number of trajectories N_{tot}. In this figure $N_{tot} = 20$ for CPU and $N_{tot} = 40$ for GPU. This data can be fitted by the exponent $N_{remain} = N_{tot}\exp(-t_{life}/t_{avr})$ where t_{avr} is the average lifetime.

Then the homogeneous crystallization rate can be calculated as $J = (t_{\text{avr}} V)^{-1}$ where V is the system volume.

As seen from Fig. 8 that the results obtained on CPU and GPU coincide within the statistical errors. Note that LAMMPS uses double precision and the higher order interpolation which has minor effect on the results for this particular problem.

These simulations were performed on the MIPT Hybrid Workstation and on the MIPT hybrid cluster which has 6 Tesla C1060 GPUs. The trajectories for different initial states were computed on different GPUs simultaneously.

7. Conclusions

The presented benchmarks show that the usage of GPUs is profitable for MD simulations of the condensed matter. The best performance was obtained by HOOMD running with single precision on the latest Nvidia GeForce 480GTX GPU which works up to 7–8 times faster than LAMMPS running with double precision on two Intel Xeon E5520 GPUs. Additional benchmarks should be performed to compare these codes for the same floating point precision.

The maximal number of particles is limited by the memory needed to store the neighbor list (10^6 per 1 GiB) which may be a problem for GPUs having less memory than the host system. In this case a more memory-efficient algorithm is to be developed for GPUs.

Nearly the same top performance ratios were obtained for both Lennard-Jones and EAM interaction models which indicates the EAM implementation discussed in the paper is quite optimal.

The results for the crystallization rate of the supercooled Al obtained on GPU and CPU agree within the statistical errors despite GPU runs on lower precision and uses less accurate interpolation algorithm. The method of averaging over the ensemble of microscopically different MD trajectories provides the possibility of efficient usage of multi-GPU workstations, GPU clusters and Grid systems.

The analysis of the obtained crystallization rate values and their comparison with the results of other approaches will be published elsewhere.

Acknowledgements

We acknowledge A.S. Kholodov, Ya.A. Kholodov, I.N. Groznov and the administrations of MIPT and JIHT RAS for access to the computational systems. The work is supported by the Programs of Fundamental Research of RAS No. 12 (coord. by G.I. Savin) and No. 14 (coord. by S.V. Emelyanov, Yu.I. Zhuravlev), RF President Grants No. MK-64941.2010.8 and MK-6719.2010.8, RFBR Grant No. 09-08-12161-ofi_m.

References

[1] A.Y. Kuksin, G.E. Norman, V.V. Stegailov, A.V. Yanilkin, P.A. Zhilyaev, Int. J. Fract. 162 (2010) 127.
[2] A.Y. Kuksin, I.V. Morozov, G.E. Norman, V.V. Stegailov, I.A. Valuev, Mol. Simulat. 31 (2005) 1005.
[3] V.V. Stegailov, A.V. Yanilkin, J. Exp. Teor. Phys. 104 (2007) 928.
[4] T.T. Bazhirov, G.E. Norman, V.V. Stegailov, J. Phys.: Condens. Mat. 20 (2008) 114113.
[5] V.V. Stegailov, Comput. Phys. Commun. 169 (2005) 247.
[6] S.V. Starikov, V.V. Stegailov, Phys. Rev. B 80 (2009) 220104(R).
[7] S.J. Plimpton, J. Comput. Phys. 117 (1995) 1.
[8] J.E. Stone, D.J. Hardy, I.S. Ufimtsev, K. Schulten, J. Mol. Graph. Model (2010) 1, doi:10.1016/j.jmgm.2010.06.010.
[9] J.A. Anderson, C.D. Lorenz, A. Travesset, J. Comput. Phys. 227 (2008) 5342.
[10] J.A. van Meela, A. Arnolda, D. Frenkelb, S.F.P. Zwartc, R.G. Belleman, Mol. Simulat. 34 (2008) 259.
[11] M.S. Friedrichs, et al., J. Comp. Chem. 30 (2009) 864.
[12] http://developer.nvidia.com/object/gpucomputing.html.
[13] M.S. Daw, S.M. Foiles, M.I. Baskes, Mater. Sci. Rep. 9 (1993) 251.
[14] Y. Mishin, D. Farkas, M.J. Mehl, D.A. Papaconstantopoulos, Phys. Rev. B 59 (1999) 3393.
[15] M.I. Mendelev, M.J. Kramer, C.A. Becker, M. Asta, Philos. Mag. 88 (2008) 1723.
[16] M. Teruaki, M. Shinji, Phys. Rev. B 69 (2004) 073307.