Domain-decomposition Parallelization for Molecular Dynamics Algorithm with Short-ranged Potentials on Epiphany Architecture

Vsevolod Nikolskii[1, 2, *] **and Vladimir Stegailov**[1, 2, **]

(Submitted by A. V. Lapin)

[1] *National Research University Higher School of Economics, 20 Myasnitskaya ul., 101000 Moscow*
[2] *Joint Institute for High Temperatures of Russian Academy of Sciences, 13 bldg 2 Izhorskaya ul., 125412 Moscow*

Received June 21, 2018

Abstract—Many-core processor architecture is a promising paradigm for the development of modern supercomputers. In this paper, we consider the parallel implementation of the generic molecular dynamics algorithm for the many-core Epiphany architecture. This architecture implements a new type of many-core processor composed of 16 simple cores connected by a network on chip with mesh topology. New approaches to parallel programming are required to deploy this processor. We use LAMMPS running on one 64-bit ARMv8 Cortex-A53 CPU core for comparing the accuracy of the results of the presented variant of the molecular dynamics algorithm for Epiphany and its computational efficiency.

2010 Mathematical Subject Classification: 68-04

Keywords and phrases: *Atomistic modelling, Lennard-Jones potential, OpenSHMEM, Epiphany, optimization.*

1. INTRODUCTION

Molecular dynamics (MD) is an extremely powerful mathematical and computational tool of modern science. MD models are used in materials science, chemistry, biology, physics and many interdisciplinary fields. Users of the method perform researches to refine the models, to achieve a better fit to experimental data, to expand the limits of applicability of the method, and to create new empirical interaction potentials. However, this paper does not concern these topics directly, it is devoted to the computational aspects of the molecular dynamics method.

Since MD is a very computationally demanding problem and it accounts for a large fraction of the computational time on the supercomputers all over the world, the issues of effective implementation and parallelization techniques of the method are well studied. Nevertheless, these issues are closely related to the particular considered computer architecture.

The possibilities of using MD calculations to solve real problems are significantly limited by the achievements of modern computer industry. To solve a number of urgent problems, at least the exaflop level of computing power is required, the achievement of which is associated with many difficulties.

After many years of extensive growth, the dominant computer architecture has come close to its limits, and the further development of the industry lies in the use of new architectures. The many-core mass-parallel processor architecture is considered as a promising technology [1]. Among modern devices, Epiphany is almost the only available for a wide range of researchers example of mass-parallel processor architecture and deserves close attention [2].

In this paper we consider the adaption of MD algorithm for parallel processor architecture Epiphany. Over the naive implementation of the classical molecular dynamics code with a short-range potential, we describe a method for reducing the exchange between processors in a parallel program.

* E-mail: vnikolskiy@hse.ru
** E-mail: v.stegailov@hse.ru

2. RELATED WORK

Development of new algorithms for novel many-core processor architectures becomes more and more complicated as the degree of parallelism increases (see e.g. [3]). At the same time, the search for energy efficient novel architectures is an important trend [4, 5].

The balance of programming complexity for data-parallel accelerators was discussed by Lee et al. [6]. In the recent review [7], the key aspects of accelerator-based systems performance modelling were considered. Wu et al. revealed the properties of MD codes on multi- and many core processors [8].

The work [9] was devoted to the d evelopment of general-purpose HPC libraries for the Epiphany architecture. Ross and Richie discussed a threaded MPI model and its implementation for Epiphany [10]. The design of the OpenMP 4.0 infrastructure for the Parallella board was presented in [11].

Sukhinov and Ostrobrod [12, 13] reported a successful implementation of an applied face-detection algorithm for the Epiphany-III coprocessor. The paper [14] discusses the use of the Parallella board with E16G3 for solving the problem of computational fluid dynamics. A simple test program was implemented, performance was measured and compared with a modern server processor and graphics accelerator. At a low overall performance, the Parallella platform showed high energy efficiency comparable to a graphics accelerator. In the paper it was shown that the small amount of memory available on the computing elements is a serious limitation for the algorithm. Thus, the results obtained in the work are characteristic of a particular class of algorithms, and can not be directly generalized to the molecular dynamics.

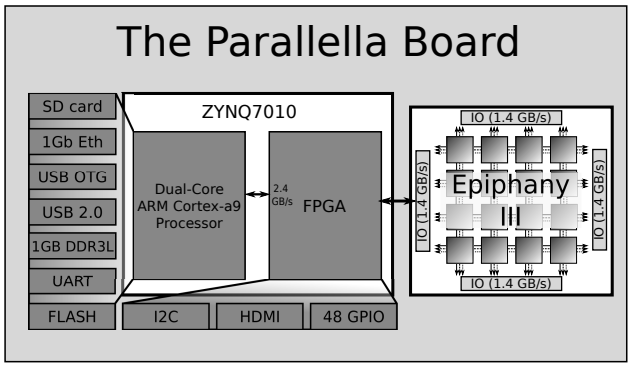

Figure 1. The scheme of the Parallella board

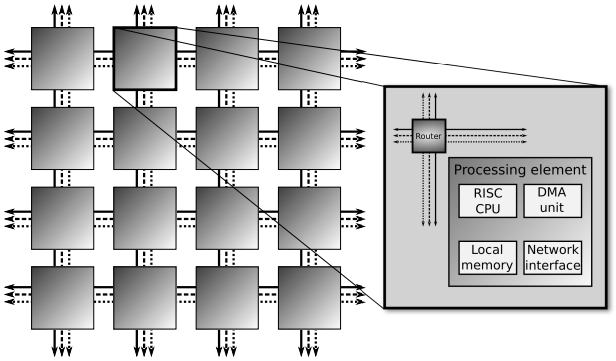

Figure 2. The Epiphany chip architecture scheme. Each core includes one RISC CPU, 32 KB of fast local memory, DMA engine and the connection to the fast on-chip network with 2D mesh topology

3. EPIPHANY ARCHITECTURE AND PROGRAMMING MODEL

In this work, we use the prototype board Parallella (Fig. 1). It includes a dual-core ARM host CPU, FPGA and a 16-core Epiphany-III co-processor (E16G301) and 1GB SDRAM. There are several interfaces: Gigabit Ethernet, Micro-SD storage, 48 GPIO pins, HDMI and USB. The host runs an Ubuntu Linux modification (so-called Parubuntu Linux).

The Epiphany architecture [15] is a distributed shared memory architecture comprised of an array of RISC processors communicating via a low-latency mesh network on chip (NoC), see Fig. 2. The eMesh NoC consists of three separate and orthogonal mesh structures, each serving different types of transaction traffic.

1. The cMesh is used for write transactions to on-chip mesh-nodes. It has a maximum bandwidth of 4.8 GB/s up, and 4.8 GB/s down in each of the four routing directions. Write transactions move through the network with a latency of 1.5 clock cycles per routing hop. A transaction traversing from the left edge to right edge of a 64-core chip would thus take 12 clock cycles.

2. The rMesh is used for all read transactions. Read transactions do not contain any data, but instead travel across the rMesh until the destination node is reached. Here, a write transaction is initiated to transport the data back to the requesting node. The rMesh can issue one read transaction every 8 clock cycles, resulting in 1/8th of the maximum cMesh bandwidth.

3. The xMesh is used for write transactions destined for off-chip resources and for passing through transactions destined for another chip in a multi-chip configuration. It is split in a North-to-South and an East-to-West network. The bandwidth of the xMesh is matched to the off-chip links of the architecture.

Each node in the processor array is a complete RISC processor capable of running an operating system with small amount of fast local memory (32 KB).

Epiphany uses a flat cacheless memory model. All amount of the distributed memory is readable and writable by all processors in the system. The edges of the 2D array can be connected to non-Epiphany interface modules, such as memory modules, FIFOs, I/O link ports, or standard buses. The array of processors with 32-bit address map can be scaled up to 4095 cores on a single chip. The existing prototype Epiphany-V reaches the value of 1024 cores on a single chip [16]. It is able to demonstrate 50–70 GFlops/W processing efficiency at the core supply level through such architectural properties as absence of cache. According to Vocke, E16G301 peak power efficiency of 32 GFlops/W can be attained at 400 MHz clock frequency [17], while on the Parallella board the Epiphany coprocessor runs at a fixed frequency of 600 MHz.

In this work, the ARL OpenSHMEM for Epiphany is used for the parallel algorithm development [18, 19]. This is an open source OpenSHMEM 1.4 implementation that can be built using Epiphany eSDK.

4. MOLECULAR DYNAMICS MODEL

The dynamics of N interacting particles is described by the system of equations

$$m_i \frac{d^2\mathbf{r}_i}{dt^2} = \mathbf{F}_i(\mathbf{r}_i, \ldots, \mathbf{r}_N), \quad \text{or} \quad m_i \frac{d\mathbf{v}_i}{dt} = \mathbf{F}_i(\mathbf{r}_i, \ldots, \mathbf{r}_N), \frac{d\mathbf{r}_i}{dt} = \mathbf{v}_i,$$

where m_i, $\mathbf{r}_i$ and $\mathbf{v}_i$ are the mass, coordinate, and velocity of the i-th particle. Force $\mathbf{F}_i$, acting on it is defined as

$$\mathbf{F}_i(\mathbf{r}_i, \ldots, \mathbf{r}_N) = -\frac{\partial U(\mathbf{r}_i, \ldots, \mathbf{r}_N)}{\partial \mathbf{r}_i}.$$

The potential function U determines the physical properties of the system. In this work, we use the Lennard-Jones potential, which represents the generic interaction of neutral atoms:

$$U(r) = 4\varepsilon \left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6} \right].$$

In computer simulations, the Lennard-Jones potential can be considered equal to zero for sufficiently long distances (e.g., $r \geq 2.5\sigma$). We use such a truncated potential in this work.

The integration of equations of motion is performed by the velocity Verlet scheme

$$\mathbf{v}_i(t + \frac{\Delta t}{2}) = \mathbf{v}_i(t) + \frac{\mathbf{F}_i(t)}{m_i}\frac{\Delta t}{2}, \tag{1}$$

$$\mathbf{r}_i(t + \Delta t) = \mathbf{r}_i(t) + \mathbf{v}_i(t + \frac{\Delta t}{2})\Delta t, \tag{2}$$

$$\mathbf{v}_i(t + \Delta t) = \mathbf{v}_i(t + \frac{\Delta t}{2}) + \frac{\mathbf{F}_i(t + \Delta t)}{m_i}\frac{\Delta t}{2}. \tag{3}$$

This scheme is well studied. The optimal properties of the scheme for molecular-dynamics simulations was shown [20, 21].

```
Initial Setup
Periodic Boundary Conditions
Compute Forces

loop over N time steps:
    Verlet Initial Integrate
    Periodic Boundary Conditions

    Clear Forces
    Compute Forces
    Verlet Final Integrate
           (a)
```

```
*Initial Setup
Periodic Boundary Conditions
*Compute Forces

loop over N time steps:
    Verlet Initial Integrate
    Periodic Boundary Conditions
    *Particles exchange
    Clear Forces
    *Compute Forces
    Verlet Final Integrate
           (b)
```

Figure 3. The pseudocode of the main loop of the MD program: (a) the atom decomposition parallelization; (b) domain decomposition the parallelization.

5. IMPLEMENTATION

5.1. Program structure

The conceptual scheme of an MD simulation program is presented on Fig. 3. The Verlet scheme is separated in two steps: `Verlet Initial Integrate` that corresponds to formulas (1), (2) and

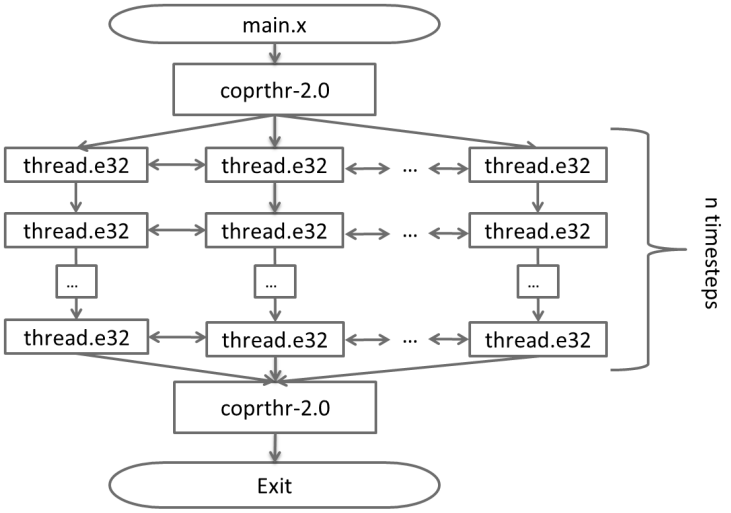

Figure 4. The scheme of the threaded host-device (CPU – Epiphany) program for the Parallela board

`Verlet Final Integrate` that corresponds to (3). Between these steps we update forces. This is the most intensive part of the algorithm (it costs about 80% of the total computational time).

The difference between two algorithms on Fig. 3 is in the approaches to parallelism. In the details, all functions are different in these two cases, but the key algorithmic differences are in the steps "Initial Setup", "Compute Force" and "Particles exchange" (highlighted by asterisk).

The current state of an MD model is represented as two arrays of structures. Since Epiphany is a cacheless processor, it does not matter whether one use an array of structures or a structure of arrays.

The fist structure contains 3 coordinates (a three-dimensional vector) and the ID of a particle, which are needed to compute the interactions. The second structure contains 3 velocities and 3 forces, that are necessary for the evolution of the system. In single precision, it gives 40 bytes per particle. The 32 KB of local memory in each core contains syscore, typically 4–20 KB of program code, the stack and free memory, available by the OpenSHMEM memory management routines. The memory management routines are atypical on Epiphany [18]. Theoretically, not more than 600 particles fit in 24 KB of memory per core of Epiphany. In practice, the program is stable up to 250 particles per core.

During the "Initial Setup" step (before the main loop) the MD data is loaded from the main global memory to the local memory of each core of Epiphany. In the case of atom decomposition approach, data for all the particles in the MD model are equally divided between the local core memory blocks and remain there until the end of the calculation. In the case of domain decomposition approach, each particle takes place in the memory of a core according to its coordinate in the MD simulation box. To maintain this state, a "Particles exchange" communication is performed on each time step. The force computation is adapted to the parallelization approach in both cases (see Algorithm 1 and Algorithm 2).

```
// Number of particles is predefined and the same on each processing element
const n = Total num. of particles / num. of PEs;
forall the processing elements of Epiphany do in parallel
    my_ca = array of n particles coordinates vectors from this PE;
    my_fa = array of n particles forces vectors from this PE;
    forall the PE of Epiphany do
        Select PE;
        remote_ca = RDMA to coordinates vectors on selected PE;
        for i = 0 to n do
            r1 = my_ca[i];
            foreach r2 in remote_ca do
                distance = |r1 - r2|;
                if distance ≤ rc^2 then
                    f = PairForce(distance);
                    my_fa[i] += f · (r1 - r2);
                end
            end
        end
    end
end
```

Algorithm 1: The parallel force computation loop in the case of atom decomposition approach

5.2. Parallelism

As it was mentioned above, the force computation loop takes about 80 % of the total time to solution, thus the most of the effort is devoted to accelerating this part of the MD algorithm. Fortunately, the natural parallelism in MD is that the force calculations and velocity/position updates can be done simultaneously for all atoms [22]. To do this, the calculated equations must be distributed among the processors. It is achieved in two popular ways, both of which are discussed in more details below. The analysis of parallelism is limited by the following conditions:

```
forall the processing elements of Epiphany do in parallel
    // Num.  of particles varies on PEs and changes over time
    n = num. of particles on this PE;
    my_ca = array of particles coordinates vectors from this PE;
    my_fa = array of particles forces vectors from this PE;
    forall the PE neighboring to this PE do
        Select PE;
        remote_n = number of particles on selected PE;
        remote_ca = RDMA to coordinates vectors on selected PE;
        for i = 0 to length(my_ca) do
            r1 = my_ca[i];
            foreach r2 in remote_ca do
                distance = |r1 - r2|;
                if distance ≤ rc^2 then
                    f = PairForce(distance);
                    my_fa[i] += f · (r1 - r2);
                end
            end
        end
    end
end
```

Algorithm 2: The parallel force computation loop in the case of domain decomposition approach

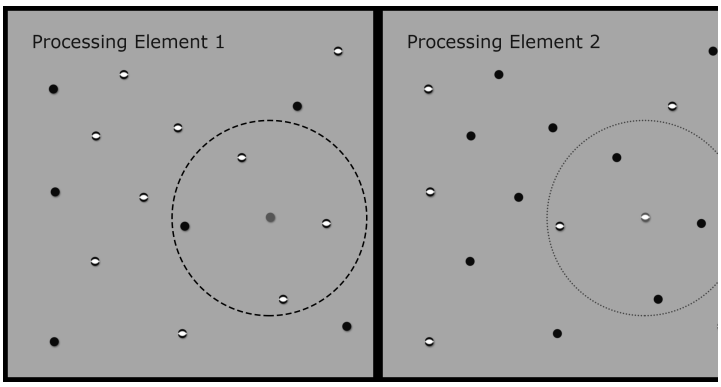

Figure 5. The atom decomposition scheme: an example for the case of two cores. Each box is a representation of the whole computational domain in the memory of a core. The particles that are stored in the local memory of some core are shown as filled circles. Particles that are accessed by the remote core via the network-on-chip interconnect are shown as open circles. The potential cutoff radius in depicted around the same particle on both cores

1. The small amount of memory on a single core. We can not test a whole medium size problem on a single core of Epiphany, the data must be "spread out" throughout the entire computational field to solve the problem.

2. Only 16 cores are available the Epiphany-III chip that is used in this work.

It will be shown below that under such conditions the issue of parallelism in our case is closely related to the algorithms of finding all atom pairs.

5.2.1. Atom decomposition. All particles are distributed among the cores, regardless of their geometric positions in the model. Every core get a subgroup of atoms (Fig. 5), and processor computes forces on its atoms no matter where they move in the course of the MD simulation.

At an every time step, "all-to-all" data exchanges are performed to search and receive coordinates of neighbor particles, because interacting particles (i.e. located closely enough in the simulation box at this time step) can be found on any other core. This communication provides not a significant load

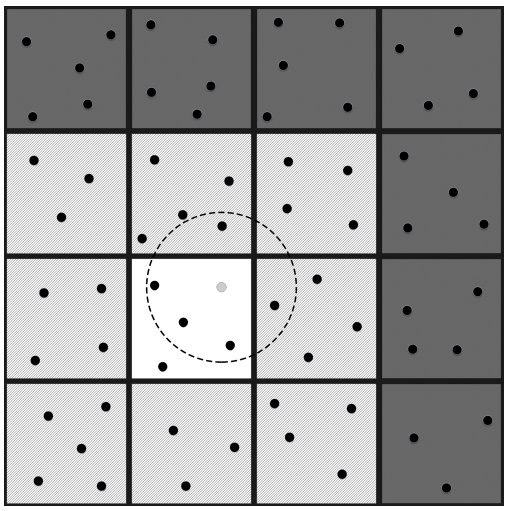

Figure 6. The domain decomposition scheme. The white square is an area dedicated to a single core. The light-gray squares represent the adjacent cores that contain the particles that can be located closely enough to interact with particles on the core considered. The particles in the dark-gray area are too far from the considered domain to contribute to the interparticle interactions. The circle represents a cut-off radius of the potential.

for the 16-cores Epiphany chip with very fast and low-latency NoC, but massive and frequent "all-to-all" communications are a limiting factor for scalable algorithms. Thus they must be eliminated.

This approach is quite easy to implement on Epiphany with shared memory. While one has to store identical copies of atoms information on all cores in a distributed memory system, information replication is not required while using shared memory. On each time step, the atom information from other processors is obtained by direct memory access in the force computation loop.

5.2.2. Domain decomposition. The MD simulation box is divided into blocks and each block is assigned to one of the processors cores. All particles from a certain block are stored in the memory of the corresponding core. As a particle moves through the MD simulation box, it passes to another core. It is done in the "Particles exchange" part on each time step that is presented on Fig. 3 and discussed in previous subsection 5.5.1. To calculate the interactions for particles on each core, it is sufficient to make exchanges not with all cores but to communicate only with neighboring cores to cover the cut-off radius of the potential (Fig. 6). It is shown in the comparison of Algorithm 1 and Algorithm 2.

The benchmarking of two decomposition techniques and their comparison with popular MD code LAMMPS [22] are represented on Fig. 7. The test model was configured for constant volume that minimize difference between the cut-off radius r_c and the domain decomposition block edge length d. The number of atoms was varied by the change of density. The MD package LAMMPS was run on single core of ARMv8 Cortex-A53 processor in double precision. It saves all data in the main memory and thus have no parallelization overhead. On the other hand, Parallella has higher peak performance and Epiphany-III uses single precision floating point arithmetic, so LAMMPS timings should not be compared taking into account the differences of the peak performance. Loop unrolling is very beneficial for acceleration of computation on Epiphany cores, however in this case more local memory on each core is needed for the code itself and the maximum number of particles in the MD model becomes lower.

The peak floating point performance of Epiphany (in single precision) is 19.2 GFlops that is 4 time higher than the peak floating point performance (in double precision) of one Cortex-A53 core considered. Our MD algorithm in single precision on Epiphany is 2.5 time faster on Epiphany than the similar algorithm in LAMMPS running on a Cortex-A53 core. The non-ideal scaling with respect to the peak floating point performance can be explained by the memory access limitation on the Epiphany architecture.

5.3. Interference of parallelism and complexity reduction

There are two most common approaches to reduce the N^2 complexity of N-body problems with short-range potentials: the Verlet neighbor list method and the cell lists method. In modern MD

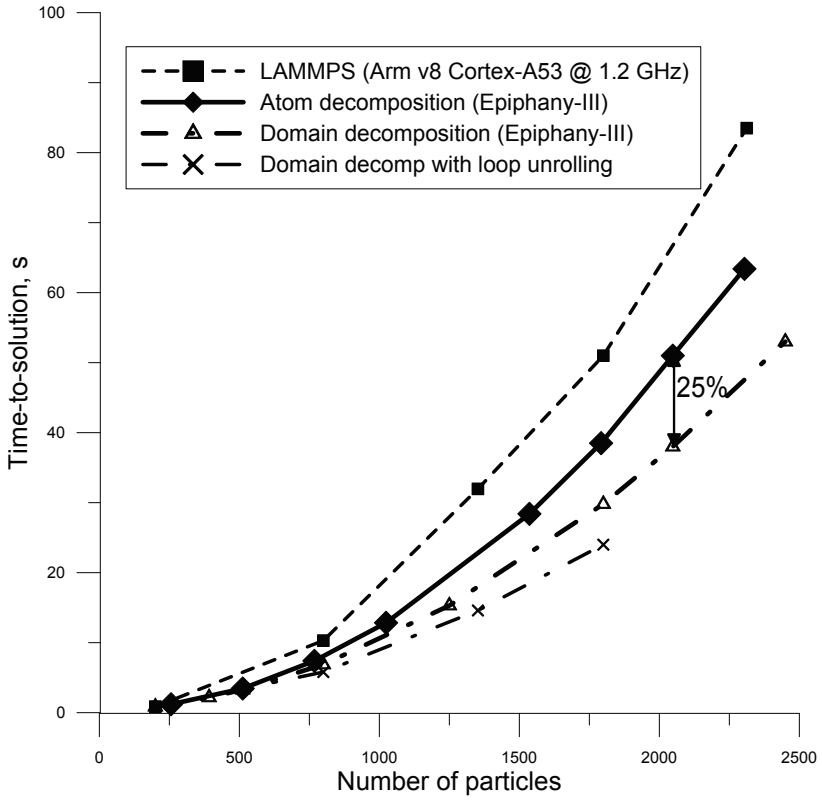

Figure 7. The time-to-solution plotted versus the number of particles in the model. We present results for atom decomposition and domain decomposition run on Parallella with Epiphany-III chip and the performance of LAMMPS package on single core of ARMv8 Cortex-A53 core (Raspberry Pi 3)

packages, the combination of these two methods is used usually to achieve better performance. Neighbor lists require huge amount of extra memory, thus they are not applicable on Epiphany due to strict memory limitations. Cell lists are implemented for the Epiphany MD code in the framework of this study. For the atom decomposition parallelization method, the use of the cell lists for particles on all cores simultaneously is not effective due to the low number of particles on separate cores.

There is a reasonable relation between the parameters of the LJ potential, the cut-off radius and the density of particles in the MD model. The number of particles that fits into the memory of a single core is also given. In this way, the range of the most frequent block edge lengths (d) in the domain decomposition method is determined.

In the case of $d \gg r_c$, it is effective to implement separate complexity reduction algorithm. In the case of Epiphany, d is relatively close to r_c, and the division of particles into the cells is naturally maintained by the domain decomposition algorithm. If we implement both domain decomposition and cell linked-list algorithms, we have to pay a full cost for the latter in terms of computer time, while it does not bring much time saving.

Without additional cell lists on every time step, a core just gets the information only from itself and from nearest cores (e.g. for 2D projection there are 8 neighbour cores, Fig. 6). In this way, instead of computation of $N * N$ pair forces, we reduce this number to $N * (N * 9/16)$. However, it still has non-linear complexity, that is shown of Fig. 7. But for the given configuration, it is more effective than the classic close-to-linear algorithm with a much higher time-to-solution.

Since in our configuration we have very few data exchanges, the key difference in time-to-solution between atom-decomposition and domain-decomposition comes from reduction of number of distance computations due to the linked-cell mechanics naturally included in our domain-decomposition implementation.

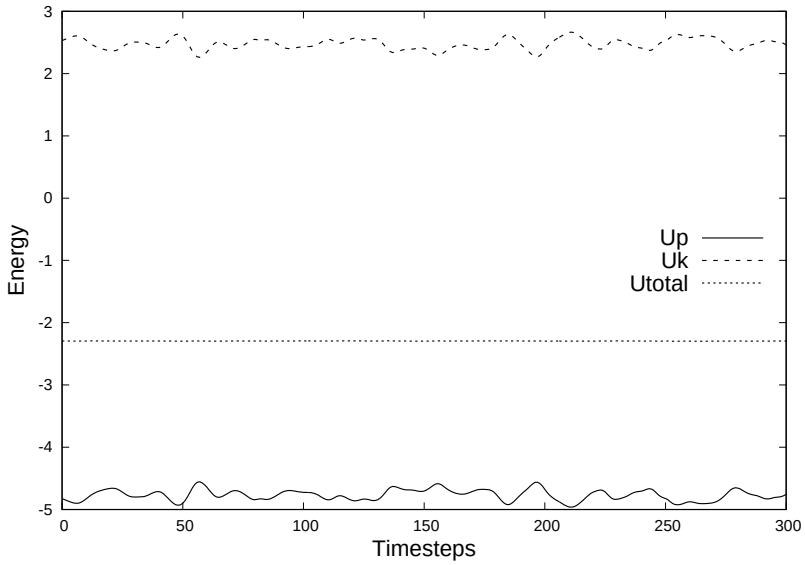

Figure 8. The conservation of the total energy during the evolution of the system

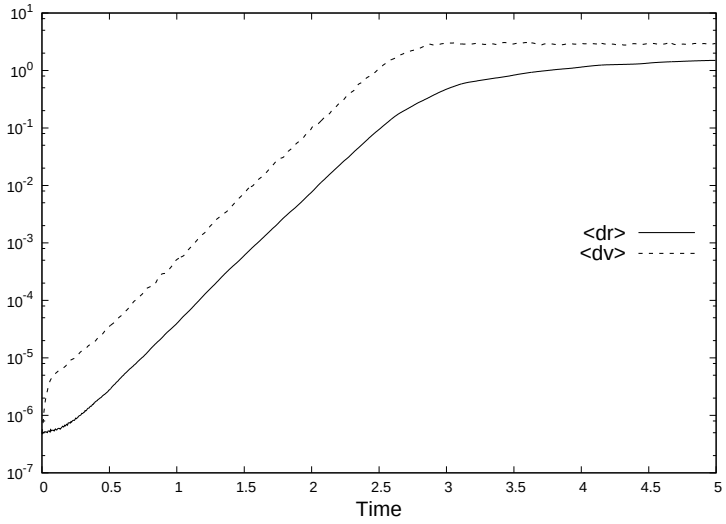

Figure 9. The normalized averaged divergences of coordinates and velocities on two trajectories calculated from identical initial conditions with LAMMPS and with our code. The exponential dependence with a further saturation regime is well explained by the stochastic theory of molecular dynamics [23, 24] and confirms the correctness of the program

5.4. Verification

The verification of our prototype program is one of the most important steps in the development. We used few simple criteria:

1. A constant level of the average total energy:

$$U_p = \sum_{i<j} U(\mathbf{r}_i, \mathbf{r}_j), \quad U_k = \sum_i \frac{m_i \mathbf{v_i}^2}{2}, \quad U_{\text{total}} = U_k + U_p \approx const.$$

In MD simulation, the total energy oscillates with some acceptable accuracy (Fig. 8). This criteria is very basic, but it is quite sensitive and allow to detect many mistakes.

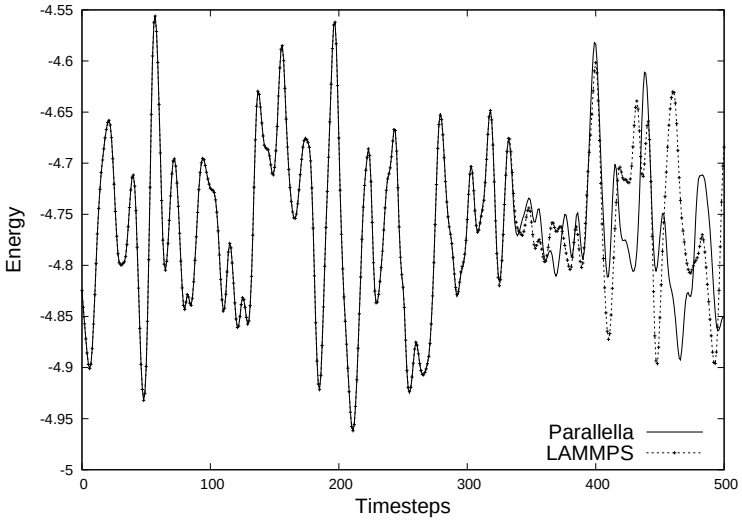

Figure 10. The exponential divergence in time of the averaged potential energy calculated by LAMMPS and by our code

2. The comparison of the potential energy time evolution with an *a priory* correct program results (we use the popular package LAMMPS [22]). The potential energy is a function of all particles positions and at the same time it represents a macroscopic value. It shows a perfect agreement for a short period of time (Fig. 10) but then diverges according to stochastic theory of MD [23].

3. The direct comparison of the resulting coordinates or velocities of atoms with the coordinates or velocities, calculated by the reference program with the same initial conditions and the same time step (Fig. 9):

$$< \Delta r(t) >= \frac{1}{N}\sum_{i=1}^{N}(\mathbf{r_i}(t) - \mathbf{r'_i}(t))^2, \quad < \Delta v(t) >= \frac{1}{N}\sum_{i=1}^{N}(\mathbf{v_i}(t) - \mathbf{v'_i}(t))^2.$$

Once we meet these criteria, one may be sure that the new simulation code implements the same mathematical model as the reference code. The exponential divergence of trajectories is the well-studied effect associated with numerical instability of the molecular dynamics trajectories.

By default, LAMMPS performs calculation in double precision floating-point arithmetic, while Parallella with Epiphany-III supports only single precision hardware accelerated arithmetic. Hardware double precision is implemented in the newer models of Epiphany only. Single precision MD is implemented in most packages (e.g. LAMMPS, GROMACS, HOOMD). Single precision is sufficient for MD simulations. It is especially useful for calculations on desktop-level GPUs, which have limited double-precision performance. That is why the Epiphany-III chip limitation of floating point operations in single precision only is not crucial for the MD algorithm.

6. CONCLUSIONS & FUTURE WORK

We described the OpenSHMEM implementation for the Epiphany architecture of the domain-decomposition parallelization for a generic molecular dynamics algorithm with the short-ranged Lennard-Jones potential. The correctness of the new algorithm was verified by the comparison with the same model calculation with LAMMPS. The difference between the resulting trajectories corresponds to the machine precision. It was shown that manual loop unrolling speeds up algorithm significantly. The comparison with LAMMPS running on a single ARMv8 Cortex-A53 core shows that the algorithm for Epiphany running on all 16 cores is 2.5 times faster.

Despite being very simple, the Epiphany processor can be considered as a prototype of future CPUs for exascale systems. For example, Epiphany has similarities with the SW26010 processor

of the Sunway TaihuLight supercomputer. That is why the development of parallel algorithms for the Epiphany architecture can be considered in a wider perspective of the general development of future supercomputer technologies.

Acknowledgments. The study has been funded by the Russian Science Foundation (grant No. 14-50-00124).

REFERENCES

1. J. A. Ang, R. F. Barrett, R. E. Benner, D. Burke, C. Chan, J. Cook, D. Donofrio, S. D. Hammond, K. S. Hemmert, S. M. Kelly, "Abstract machine models and proxy architectures for exascale computing," et al., in *2014 Hardware-Software Co-Design for High Performance Computing* (2014) 25–32.
2. W. D. Gropp, "MPI+X for Extreme Scale Computing," in *12th Int. Conf. on Parallel Processing and Applied Mathematics* (2017).
3. B. Glinsky, I. Kulikov, I. Chernykh, D. Weins, A. Snytnikov, V. Nenashev, A. Andreev, V. Egunov, and E. Kharkov, "The co-design of astrophysical code for massively parallel supercomputers," in *Algorithms and Architectures for Parallel Processing*, edited by J. Carretero et al. (Springer International Publishing, Cham, 2016) 342–353.
4. D. D. Pruitt and E. A. Freudenthal, "Preliminary investigation of mobile system features potentially relevant to HPC," in *Proceedings of the 4th International Workshop on Energy Efficient Supercomputing* (IEEE Press, Piscataway, NJ, USA, 2016).
5. F. Mantovani and E. Calore, "Performance and power analysis of HPC workloads on heterogeneous multi-node clusters," J. of Low Power Electronics and Applications **8** (2018).
6. Y. Lee, R. Avizienis, A. Bishara, R. Xia, D. Lockhart, C. Batten, and K. Asanović, "Exploring the tradeoffs between programmability and efficiency in data-parallel accelerators," SIGARCH Comput. Archit. News **39**, 129 (2011).
7. U. Lopez-Novoa, A. Mendiburu, and J. Miguel-Alonso, "A survey of performance modeling and simulation techniques for accelerator-based computing," IEEE Transactions on Parallel and Distributed Systems **26**, 272 (2015).
8. Q. Wu, C. Yang, T. Tang, and L. Xiao, "MIC Acceleration of short-range molecular dynamics simulations," in *Proceedings of the First International Workshop on Code OptimiSation for Multi and Many Cores* (ACM, New York, NY, USA, 2013).
9. M. Tasende, "Generation of the single precision BLAS Library for the Parallella platform, with Epiphany co-processor acceleration, using the BLIS framework," in *2016 IEEE 14th Intl Conf on Dependable, Autonomic and Secure Computing, 14th Intl Conf on Pervasive Intelligence and Computing, 2nd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress(DASC/PiCom/DataCom/CyberSciTech)* 894–897 (2016).
10. J. A. Ross, D. A. Richie, S. J. Park, and D. R. Shires, "Parallel programming model for the Epiphany many-core coprocessor using threaded MPI," Microprocessors and Microsystems **43**, 95 (2016).
11. S. N. Agathos, A. Papadogiannakis, and V. V. Dimakopoulos, "Targeting the Parallella," in *Euro-Par 2015: Parallel Processing: 21st Int. Conf. on Parallel and Distributed Computing, Vienna, Austria, August 24–28, 2015, Proceedings*, edited by J. L. Träff, S. Hunold, and F. Versaci (Springer Berlin Heidelberg, Berlin, Heidelberg 2015) 662–674.
12. A. Sukhinov and G. Ostrobrod, "Efficient face detection on Epiphany multicore processor," in *Parallel Computational Technologies (PCT'2014)* (2014), vol. 3 of *Bulletin of the South Ural State University Series "Computational Mathematics and Software Engineering"*, 5–19.
13. A. Sukhinov and G. Ostrobrod, "Efficient face detection on Epiphany multicore processor," Computational Mathematics and Information Technologies **1**, 113 (2017).
14. S. Raase and T. Nordström, "On the use of a many-core processor for computational fluid dynamics simulations," Procedia Computer Science **51**, 1403 (2015).
15. A. Olofsson, T. Nordström, and Z. Ul-Abdin, "Kickstarting high-performance energy-efficient manycore architectures with Epiphany," in *48th Asilomar Conf. on Signals, Systems and Computers* (2014).
16. A. Olofsson, R. Trogan, and O. Raikhman, "A 1024-core 70 GFLOP/W floating point manycore microprocessor," in *15th Annual Workshop on High Performance Embedded Computing* (2011).
17. T. Vocke, "An evaluation of the Adapteva Epiphany Many-core architecture," Master's thesis, University of Twente/Thales (2015).
18. J. A. Ross and D. A. Richie, "Implementing OpenSHMEM for the Adapteva Epiphany RISC array processor," Procedia Computer Science **80**, 2353 (2016).
19. J. Ross and D. Richie, "An OpenSHMEM implementation for the Adapteva Epiphany coprocessor," in *OpenSHMEM and Related Technologies. Enhancing OpenSHMEM for Hybrid Environments*, edited by M. Gorentla Venkata, N. Imam, S. Pophale, and T. M. Mintz (Springer International Publishing, Cham, 2016) 146–159.

20. M. López-Marcos, J. Sanz-Serna, and J. Díaz, "Are Gauss–Legendre methods useful in molecular dynamics?," J. of Computational and Applied Mathematics **67**, 173 (1996).
21. M. A. López-Marcos, J. M. Sanz-Serna, and R. D. Skeel, "Explicit symplectic integrators using Hessian–vector products," SIAM J. on Scientific Computing **18**, 223 (1997).
22. S. Plimpton, "Fast parallel algorithms for short-range molecular dynamics," J. Comp. Phys. **117**, 1 (1995).
23. G. E. Norman and V. V. Stegailov, "Stochastic theory of the classical molecular dynamics method," Mathematical Models and Computer Simulations **5**, 305 (2013).
24. S. Stoddard and J. Ford, "Numerical experiments on the stochastic behavior of a Lennard-Jones gas system," Phys. Rev. A. **8**, 1504 (1973).