Energy Efficiency of Epiphany Many-Core Architecture for Parallel Molecular Dynamics Calculations

Vsevolod Nikolskii and Vladimir Stegailov
International Laboratory for Supercomputer Atomistic Modelling and Multi-scale Analysis
National Research University Higher School of Economics
Moscow, Russia
E-mail: {vnikolskiy, v.stegailov}@hse.ru

***Abstract*—The paper considers the performance and energy consumption of Parallella board with Epiphany coprocessor for molecular dynamics simulation. The coprocessor has cacheless many-core architecture, which is a promising energy efficient technology for the evolution of modern supercomputers. The paper describes the development and verification of molecular dynamics simulation program for the new platform. It reveals the capabilities of effective parallelization of the code on currently available system taking into account the future development. Comparison of the energy consumption with a modern general-purpose processor Cortex-A53 shows the advantage of the Parallella platform, while there are still opportunities to improve the software.**

Keywords–PGAS; OpenSHMEM; atomistic modelling; Lennard-Jones; n-body problem; power consumption.

I. 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 the 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. 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 [1].

The rest of this paper is organized as follows: Section II is a review of related work. In Section III we describe hardware and software system, used in this paper. Section IV briefly describes the test simulation problem. In Section V we consider the adaption of MD algorithm for parallel processor architecture Epiphany. Over the naive implementation, we describe a method for reducing the memory exchanges between processors in a parallel program. Power of the board running MD simulation is measured using digital watt-meter in Section VI. The results are compared with modern general-purpose Central Processing Unit (CPU), that have compared power. Finally, Section VII contains the conclusion.

II. RELATED WORK

The balance of programming complexity for data-parallel accelerators was discussed by Lee et al. [2]. In the recent review [3], 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 [4]. Paper [5] present the results of experiments with parallel algorithms (including MD) on Tilera's TilePro64, that shows the advantage of cashless mode.

The recent work [6] is devoted to the development of general-purpose high-performance computing libraries for the Epiphany architecture. Ross and Richie discussed a threaded Message Passing Interface (MPI) model and its implementation for Epiphany [7]. The design of the OpenMP 4.0 infrastructure for the Parallella board was presented in [8].

Sukhinov and Ostrobrod [9] reported a successful implementation of an applied face-detection algorithm for the Epiphany-III coprocessor. The paper [10] 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.

III. EPIPHANY ARCHITECTURE AND PROGRAMMING MODEL

In this work, we use the prototype board Parallella (Figure 1). It includes a dual-core ARM host CPU, FPGA (Field-Programmable Gate Array) and a 16-core Epiphany-III coprocessor (E16G301) and 1 GB of memory. There are several

 ISBN: 978-1-61208-655-2

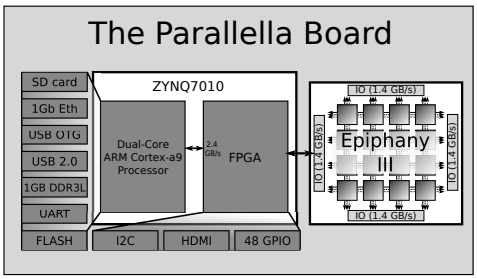

Figure 1. The scheme of the Parallella board.

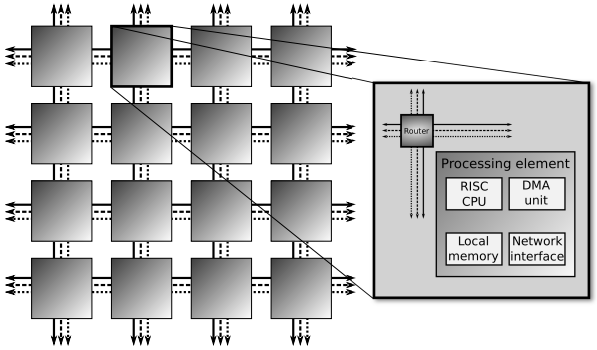

Figure 2. The Epiphany chip architecture scheme.

interfaces: Gigabit Ethernet, Micro-SD storage, 48 General-Purpose Input/Output (GPIO) pins, High Definition Multimedia Interface (HDMI) and Universal Serial Bus (USB). The host runs an Ubuntu Linux modification (so-called Parubuntu Linux). The Epiphany architecture [11] is a distributed shared memory architecture comprised of an array of Reduced Instruction Set Computer (RISC) processors communicating via a low-latency mesh Network on Chip (NoC), see Figure 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 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 [12]. Epiphany-IV (2011) with 64 cores is able to demonstrate 70 GFlops/W processing efficiency at the core supply level through such architectural properties as the absence of cache. According to Vocke, E16G301 peak power efficiency of 32 GFlops/W can be attained at 400 MHz clock frequency [13], while on the Parallella board the Epiphany co-processor runs at a fixed frequency of 600 MHz.

In this work, the OpenSHMEM for Epiphany is used for the parallel algorithm development [14][15]. This is an open source OpenSHMEM 1.4 implementation that can be built using Epiphany eSDK.

OpenSHMEM is responsible for data exchange between Processing Element (PE) and implements the parallel programming model named “Partitioned Global Address Space” (PGAS). All the memory on the processing elements is addressable, but it is divided into logical sections and allows one to consider the use of data locality. This model perfectly matches the architecture of Epiphany. The technology of Remote Direct Memory Access (RDMA) to is used. The main programming idea is to create a universal function that can process memory areas from specified computational elements.

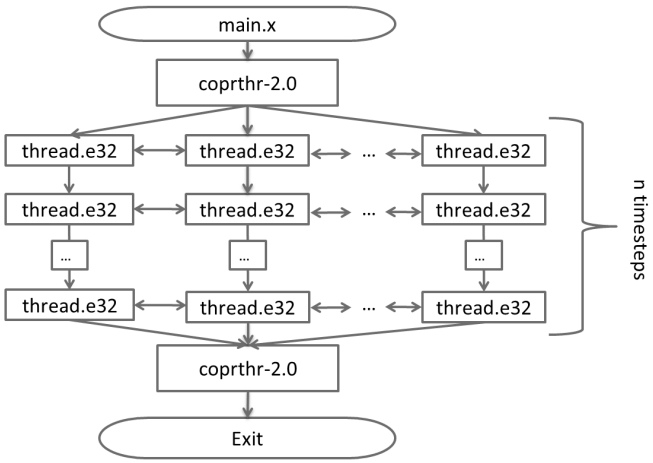

Figure 3. The scheme of the threaded host-device (CPU - Epiphany) program for the Parallela board.

Library and toolchain COPRTHR-2.0 are responsible for the loading and start of the program kernel. It loads the compiled code into PEs under the control of a lightweight OS and launches it (Figure 3). Also, through the functions of this library, the initial data is transferred from the main memory of the board. Using the utility from the COPRTHR toolkit one can analyze the memory allocation in the compiled code. An essential limitation is that the compiled code takes up to 77% of the memory of PEs, syscore and fragmentation drains up to 5% of memory, so free memory is estimated at just 6608 bytes for code with manual loop unrolling and slightly more without that optimization.

 ISBN: 978-1-61208-655-2

IV. Molecular Dynamics Model

The dynamics of N interacting particles is described by the system of Newton's equations of motion. Force $\mathbf{F}_i$, acting on a particle is defined by the potential function U, which determines the physical properties of the system. In this work, we use the Lennard-Jones potential, which represents the generic interaction of neutral atoms:

$$U(r) = 4\varepsilon \left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6} \right]. \quad (1)$$

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. This scheme is well studied, the optimality of the scheme for molecular-dynamics simulations was shown [16][17].

V. Implementation

A. Program Structure

The conceptual scheme of an MD simulation program is presented on Figure 4. The Verlet scheme is separated in two steps: `Verlet Initial Integrate` and `Verlet Final Integrate`. Between these steps the forces are updated. This is the most intensive part of the algorithm (it costs about 80% of the total computational time).

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

The difference between two algorithms on Figure 4 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 the 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 uses an array of structures or a structure of arrays — that was confirmed by the experiment.

The first structure contains three 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 memory management routines are atypical on Epiphany [14]. With only 6608 bytes of free local memory on PEs (see Section III), the simulation is limited by just ~ 140 particles per PE.

```
// Number of particles is predefined and the same on each
   processing element
const n = Total num. of particles / num. of PEs;
forall 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 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
```

Figure 5. The parallel force computation loop in the case of atom decomposition approach.

```
forall 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 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
```

Figure 6. The parallel force computation loop in the case of domain decomposition approach.

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

 ISBN: 978-1-61208-655-2

force computation is adapted to the parallelization approach in both cases (see Figure 5 and Figure 6).

B. 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 [19]. 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:

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.

1) Atom Decomposition: The particles are distributed among the cores, regardless of their geometric positions in the model. Every core gets a subgroup of atoms, and processor computes forces on its atoms no matter where they move in the course of the MD simulation. At Figure 7 each box represent 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 is depicted around the same particle on both cores. At 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 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.

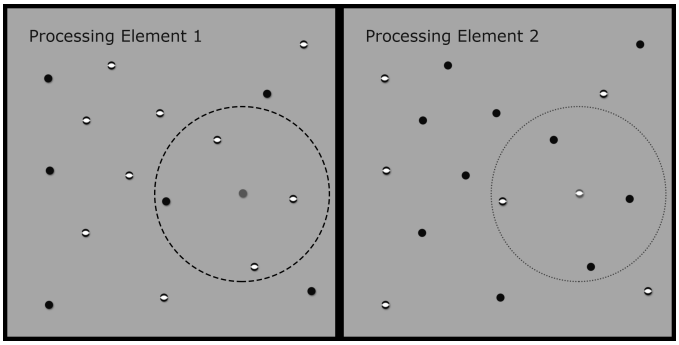

Figure 7. The atom decomposition scheme: an example for the case of two cores.

This approach is quite easy to implement on Epiphany with shared memory and hardware RDMA feature. 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.

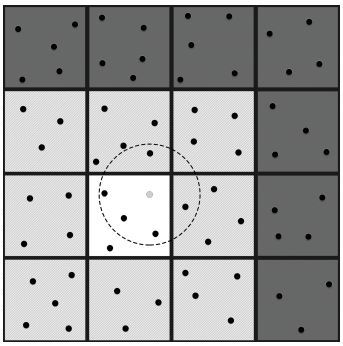

Figure 8. The domain decomposition scheme.

2) Domain Decomposition: The MD simulation box is divided into blocks and each block is assigned to one of the processor's 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 Figure 4 and discussed in previous Subsection V-A. 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. The idea is shown at Figure 8: 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 close 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.

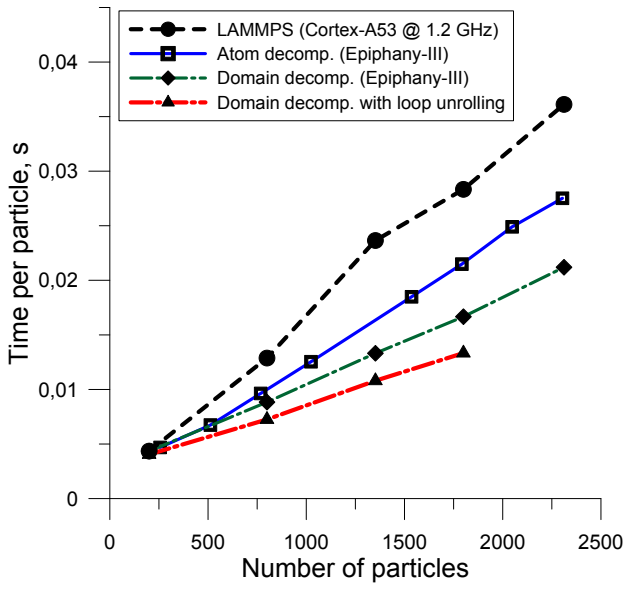

Figure 9. The time-to-solution per particle plotted versus the number of particles in the model.

The benchmarking of two decomposition techniques and their comparison with popular MD code LAMMPS [19] are represented on Figure 9, which depict the results for atom decomposition and domain decomposition run on Parallella with Epiphany-III chip and the performance of LAMMPS package on a single core of ARMv8 Cortex-A53 core. The test model was configured for constant volume that minimizes the 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 the single core of ARMv8 Cortex-A53 processor in double precision. It keeps all data in the main memory and thus has 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 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 times 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 times 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 [18] can be explained by the memory access limitation on the Epiphany architecture.

C. 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 packages, the combination of these two methods is used usually to achieve better performance. Neighbor lists require a 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 used 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 neighbor cores, Figure 8).

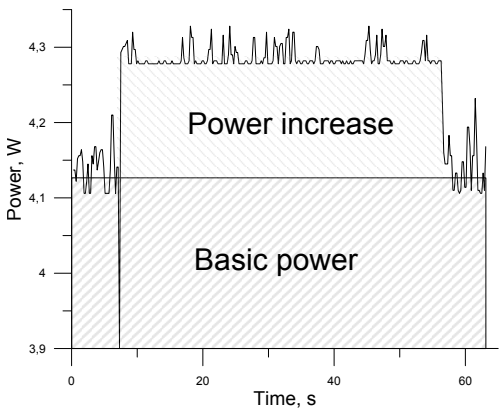

Figure 10. Energy consumption timeline.

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 Figure 9. But for the given configuration, it is more effective than the classic close-to-linear algorithm with a much higher time-to-solution.

D. Verification

The verification of our prototype program is one of the important steps in the development. We used several criteria: conservation of total energy, comparison of the potential energy time evolution with *a priory* correct program results (we use the popular package LAMMPS [19]), and 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.

By default, LAMMPS performs the 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.

VI. Energy Efficiency

To measure energy consumption, we use a high-precision digital watt-meter ODROID Smart Power. It draws an energy consumption profile with 10 Hz sample rate when the program starts. As an opponent of the Parallella, we chose the popular platform Raspberry Pi 3, since it can be supplied and measured using the same device. Peak performance of one core of Raspberry Pi 3 processor Cortex-a53 is 4 times lower than peak performance of 16 cores of Epiphany-III. Let us note that Raspberry Pi 3 is manufactured using 40 nm technology, while Epiphany-III is made using 65 nm technology. We can assume that the implementation of the same architecture in a more modern technical process can bring an additional advantage. In Figure 10 presents the energy profile when the program is started on Parallella, that can be used for direct calculation of energy-to-solution [20]. It is possible to separate

 ISBN: 978-1-61208-655-2

TABLE I. ENERGY CONSUMPTION VALUES.

	Time, s	*Total Energy, J*	*Energy Increase, J*
Parallella	24	100.7	3.8
Raspberry Pi 3	51	183.6	40.8

the background energy consumption, which mainly falls on the CPU and interfaces, from the consumption of Epiphany, which is of primary interest to us. According to our measurements, when Epiphany is under load, the power consumption increases by only 0.2 Watt. The obtained results can be used to estimate power consumption of hardware systems, running real-life supercomputing programs.

VII. CONCLUSION AND 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 times faster, while there remain opportunities for improving the algorithm. The comparison of total energy consumption shows that Parallella board is $\sim$2 times more energy efficient that Raspberry Pi 3 in terms of total energy-to-solution. The isolated consumption of Epiphany-III chip is $\sim$11 times less, than the consumption of modern and effective processor Arm Cortex-A53.

ACKNOWLEDGMENT

The study was funded by RFBR according to the research project No. 18-37-00487 and supported within the framework of the Basic Research Program at the National Research University Higher School of Economics (HSE) and within the framework of a subsidy by the Russian Academic Excellence Project 5-100.

REFERENCES

[1] W. D. Gropp, "MPI+X for extreme scale computing," in 12th International Conference on Parallel Processing and Applied Mathematics, 2017, pp. 1-38. [Online]. Available: https://www.ppam.pl/docs/presentations/Gropp_PPAM2017.pdf [accessed: 2018-06-27].

[2] Y. Lee *et al.*, "Exploring the tradeoffs between programmability and efficiency in data-parallel accelerators," SIGARCH Comput. Archit. News, vol. 39, no. 3, Jun. 2011, pp. 129–140. doi:10.1145/2024723.2000080.

[3] 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, vol. 26, no. 1, Jan 2015, pp. 272–281.

[4] 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, ser. COSMIC '13. New York, NY, USA: ACM, 2013, pp. 2:1–2:8. doi:10.1145/2446920.2446922.

[5] V. Chandru and F. Mueller, "Hybrid MPI/OpenMP programming on the Tilera manycore architecture," 2016 International Conference on High Performance Computing & Simulation (HPCS), Innsbruck, 2016, pp. 326-333. doi: 10.1109/HPCSim.2016.7568353

[6] 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, Aug 2016, pp. 894–897.

[7] 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, vol. 43, no. Supplement C, 2016, pp. 95–103, many-Core System-on-Chip: Architectures and Applications (PDP 15).

[8] S. N. Agathos, A. Papadogiannakis, and V. V. Dimakopoulos, "Targeting the Parallella," in Proceedings Euro-Par 2015: Parallel Processing: 21st International Conference on Parallel and Distributed Computing, Vienna, Austria, August 24–28, 2015, pp. 662–674. doi:10.1007/978-3-662-48096-0_51.

[9] A. Sukhinov and G. Ostrobrod, "Efficient face detection on epiphany multicore processor", Computational Mathematics and Information Technologies, vol. 1, no. 1, 2017, pp. 113–127.

[10] S. Raase and T. Nordström, "On the use of a many-core processor for computational fluid dynamics simulations," Procedia Computer Science, vol. 51, no. Supplement C, 2015, pp. 1403–1412, Int. Conf. On Computational Science, ICCS 2015. doi:10.1016/j.procs.2015.05.348

[11] A. Olofsson, T. Nordström, and Z. Ul-Abdin, "Kickstarting high-performance energy-efficient manycore architectures with Epiphany," in 48th Asilomar Conference on Signals, Systems and Computers, pp. 1719–1726, 2014. doi:10.1109/ACSSC.2014.7094761

[12] A. Olofsson, "Epiphany-V: A 1024 processor 64-bit RISC System-On-Chip," pp. 1–15, 2016, [Online]. Available: https://arxiv.org/pdf/1610.01832.pdf [accessed: 2018-06-27].

[13] T. Vocke, "An evaluation of the Adapteva Epiphany many-core architecture," Master's thesis, University of Twente/Thales, Aug 2015. [Online]. Available: http://essay.utwente.nl/68024/1/vocke_MA_EEMCS.pdf [accessed: 2018-06-27]

[14] J. A. Ross and D. A. Richie, "Implementing openshmem for the Adapteva Epiphany RISC array processor," Procedia Computer Science, vol. 80, Supplement C, 2016, pp. 2353–2356, International Conference on Computational Science (ICCS 2016), 6-8 June 2016, San Diego, California, USA.

[15] J. Ross and D. Richie, "An OpenSHMEM implementation for the Adapteva Epiphany coprocessor," in OpenSHMEM and Related Technologies. Enhancing OpenSHMEM for Hybrid Environments, Springer International Publishing, 2016, pp. 146–159.

[16] M. López-Marcos, J. Sanz-Serna, and J. Díaz, "Are Gauss-Legendre methods useful in molecular dynamics?" Journal of Computational and Applied Mathematics, vol. 67, no. 1, 1996, pp. 173–179.

[17] M. A. López-Marcos, J. M. Sanz-Serna, and R. D. Skeel, "Explicit symplectic integrators using Hessian-vector products," SIAM Journal on Scientific Computing, vol. 18, no. 1, 1997, pp. 223–238. doi:10.1137/S1064827595288085.

[18] V. P. Nikolskiy and V. V. Stegailov, "Floating-point performance of ARM cores and their efficiency in classical molecular dynamics," J. Phys.: Conf. Ser., vol. 681, pp. 012049, 2016. doi:10.1088/1742-6596/681/1/012049.

[19] S. Plimpton, "Fast parallel algorithms for short-range molecular dynamics," J Comp Phys, vol. 117, 1995, pp. 1–19. [Online]. Available: http://lammps.sandia.gov [accessed: 2018-06-27].

[20] E. Calore, S. F. Schifano and R. Tripiccione, "Energy-Performance Tradeoffs for HPC Applications on Low Power Processors," In Euro-Par 2015: Parallel Processing Workshops, Proceedings of the Euro-Par 2015 International Workshops, Vienna, Austria, 24–25 August 2015. Springer: Berlin, Germany, 2015. pp. 737–748.

 ISBN: 978-1-61208-655-2