Parallel Algorithms for Molecular Dynamics Simulation of Irradiation Effects in Crystals

ELI GLIKMAN[1], LUDMILA IOFFE[2], ITZHAK KELSON[1], AND SHLOMIT S. PINTER[2]

[1]*School of Physics and Astronomy, The Raymond and Beverly Sackler Faculty of Exact Sciences, Tel Aviv University, Tel Aviv 69978, Israel; e-mail: glikman@aristo.tau.ac.il and kelson@plato.tau.ac.il*

[2]*Department of Electrical Engineering Technion—Israel Institute of Technology, Haifa 32000, Israel; e-mail: ludmila@csc.cs.technion.ac.il and shlomit@ee.technion.ac.il*

ABSTRACT

We present new parallel algorithms for solving the problem of many body interactions in molecular dynamics (MD). Such algorithms are essential in the simulation of irradiation effects in crystals, where the high energy of the impinging particles dictates computing with large numbers of atoms and for many time cycles. We realized the algorithms using two parallelization methods and compared their performance. Experimental results obtained on a Meiko machine demonstrate that the new algorithms exploit parallelism effectively and can be used to simulate large crystals.

1 INTRODUCTION

Molecular dynamics (MD) computer simulation has been used extensively as a tool to study irradiation damage. In recent years there has been a growing interest in simulating high-energy irradiation damage using molecular dynamic calculation [2, 4].

The energetic processes (100 eV up to 30 KeV) may produce collision sequence and shock waves greater than the speed of sound. Also the initial energy, E_{init}, can raise the temperature of the crystal to several hundred kelvin, i.e., $T = E_{init}/3Nk_B$ where N is the number of atoms and k_B is Boltzmann's constant. Therefore, large computational crystals have to be used with $10^5 - 10^6$ atoms [14]. This results in a huge consumption of computing power, prohibiting, on occasion, the completion of the simulation to its physically interesting extent. To overcome this inherent difficulty, it is important to investigate new ways for reducing the computation time and for employing parallel computers.

MD methods integrate numerically the Newton equations of motion with forces derived from the potential. Computing the forces of each atom (and its state) is computationally expensive particularly when the computational crystal contains a large number of atoms. In fact, most of the computation time is spent on the calculation of the forces at every time step. A standard way to reduce the computation time is the neighbor list method. In this method, one compiles for each atom the list of neighbors which are contributing to the force acting on it. This neighbor list has to be updated periodically after a constant time interval Δt which depends on the physical problem.

In this article we introduce new parallel algorithms for calculating and managing the neighbor list; in addition, we provide a new algorithm for

Received May 1994
Revised December 1994

Scientific Programming, Vol. 4, pp. 185–192 (1995)
CCC 1058-9244/95/030185-08

computing the forces and the coordinates of an atom in a distributed manner. The first improvement is based on the fact that not all atoms have to be considered when building the neighbor list of an atom. The second improvement relies on the fact that no updates for the neighbor lists are needed at all in areas with low energy atoms.

We describe the programming techniques and present numerical results for two parallelization methods. In the first one the main data structures are partitioned between the processors and the "heavy" procedures of the sequential algorithms are parallelized whenever possible. In such an approach every processor computes only part of the values but has all the information at each time step. In this scheme every two processors exchange information. In the second scheme, the physical domain is statically divided among the processors and the algorithm is changed accordingly. In this scheme every processor communicates only with some neighboring processors to receive the required information.

In previous work, Li et al. [10] used a Transputer-based Meiko machine (no i860 processors) in a ring configuration in order to compute the forces with a parallelized version of the standard N-body algorithm.

In Section 2 we describe the application problem and the physical model used. The new algorithms are presented in Section 3. In Section 4 we discuss the parallel algorithms and the programming technique. We provide simulation and performance results of runs in Section 5 and finally, we draw our conclusions in Section 6.

2 OVERVIEW OF APPLICATION AND PHYSICAL MODEL

The MD calculation used in this work is derived from the cascade calculations in copper [12], and was adapted for treatment on nuclear-stimulated desorption in palladium [7]. The calculations are carried out on a cubic microcrystal, with a variable number of atoms. The atoms are arranged at f.c.c. lattice, and the interaction was chosen to be a many-body potential as used in earlier studies [6, 7].

Trajectories of the atoms are determined by the numerical solution of the Newtonian equations of motion. The integration scheme uses the central difference method for computing the positions and velocities of the atoms that are calculated for successive time-steps (this is known as Verlet method [13]*). Let N be the number of atoms in the microcrystallite and Δt is a given time-step. The values of the positions r_i and velocities v_i of atoms $i = 1, \ldots, N$ at time t are found by following:

$$r_i(t + \Delta t) = 2r_i(t) - r_i(t - \Delta t) + (F_i/m_i)\Delta t^2 + O(\Delta t^4). \quad i = 1, \ldots, N; \quad (1)$$

$$v_i(t) = [r_i(t + \Delta t) - r_i(t - \Delta t)]/2\Delta t + O(\Delta t^2). \quad i = 1, \ldots, N \quad (2)$$

where F_i is the force exerted on atom i of mass m_i. derived from the relations:

$$F_i = -\sum_{j \neq i} \nabla_{r_i} E_{coh}; \quad (3)$$

$$E_{coh} = \sum_i (E_i^A + E_i^R) \quad (4)$$

with

$$E_i^A = -\left(\sum_{j \neq i} \left(\xi^2 \exp(-2q\left(\frac{r_{ij}}{r_0} - 1\right)\right)\right)^{1/2}; \quad (5)$$

$$E_i^R = \sum_{j \neq i} A \exp\left(-p\left(\frac{r_{ij}}{r_0} - 1\right)\right). \quad (6)$$

In (3) ∇r_j is the gradiant operator with respect to coordinate r_j. In (5) and (6) r_{ij} is the distance between atoms i and j. and r_0 is the nearest neighbor distance; the parameters A. p. q. and ξ have been determined by experiment.

In the present method the particle potential is computed under the assumption that the atom interacts only with its neighboring atoms. The neighbors of an atom are those particles which are located within a sphere of a prescribed cutoff radius R_c around it. The cutoff R_c was chosen between the third and fourth order neighbors so as to avoid problems arising from finite temperature simulation in conjunction with short range interactions [11].

Starting from a given equilibrium configuration, one specific atom in a predetermined site is given an additional kinetic energy. The resultant cascade is then followed in phase space for 1.000 time-steps. Each run was characterized by the size of the crystal. the position of the primary en-

* There are several reasons for using this method: for more details see [1].

ergetic atom, its initial energy $E_{init} = 100$ eV. The time-step Δt was taken as $1 10^{-15}$ seconds. The total energy, the kinetic energy, and the number of displaced atoms were monitored throughout the calculation [6].

3 ALGORITHMS

The basic MD algorithm comprises three main steps: (i) computing the forces between the particles: (ii) computing the new positions of the particles: (iii) computing the global physical characteristics. Calculating the forces in step (i) consumes most of the computation time and involves the forces of neighboring atoms.

Due to the high energy processes in the irradiation damage problem, the number of particles in the computational crystal is of the order of 10^6. In this type of application the distribution of energies across the computational crystal is not homogeneous. This suggests the use of a neighbor list method (for computing the forces) which is less global in nature compared to other methods.

3.1 Existing Neighbor List Methods

The idea of the neighbor list method can be summarized as follows [13]: the neighbor list of every particle includes all the atoms that are inside the potential radius R_c.

In the Verlet method the distance to all the other atoms is calculated for each atom. Those atoms within the sphere of the radius R_c are included in the appropriate neighbor list. In this method the time for computing the neighbor list, T_{list}, is of the order of N^2. The computation of the neighbor list is done periodically in predetermined time intervals.

In the link cell method [9], the crystal is divided into a number of cells (sub-cells), whose size is determined by the potential cutoff (R_c). In this method, checking for neighbors of a given atom is limited to atoms within the same cell or in adjacent cells. Computing the list in this case is faster than in the "Verlet-list" method and is given by:

$$T_{list} \sim C \cdot n$$

where C is the number of neighbor cells multiplied by n-the number of atoms per sub-cell.

In the next section we describe two modifications of the sequential MD algorithm.

3.2 The Relational Algorithm

The relational method that we propose can be used to improve both the Verlet and the link cell methods. It is based on a recursive algorithm that uses the neighbor list at time $t - \Delta t$ to create the neighbor list at time t. The idea is to reduce the number of atoms considered during the creation of the neighbor lists. For every atom we check the neighbors in a sphere given by the potential cutoff R_c and the neighbors of these atoms. This method is based on the physical assumption that new neighbors cannot enter into the potential radius of the checked atom during a constant time-step without first becoming neighbors of some of that atom's existing neighbors. For the initial time-step we use the Verlet method to calculate the primary neighbor list. The computation time of the neighbor list is given by:

$$T_{list} \sim N \cdot (N_{fn} + N_{sn})$$

where N_{fn}, N_{sn} are the numbers of first and second order neighbors, respectively. For an f.c.c lattice, with the commonly used interaction cutoff radius, one has $N_{fn} \simeq 50$ and $N_{sn} \simeq 2^3 \cdot N_{fn}$. One advantage of the new scheme for calculating the neighbor list is its potential suitability for parallelization.

3.3 The Refreshment Algorithm

The inhomogeneous distribution of atom energies is normally such that the atoms with high energy are mainly concentrated in small regions of the crystal. On the other hand the areas of low energy atoms are not changing dramatically. Thus, there is no need to update the neighbor lists for atoms with low energy in the same frequency as for the neighbor lists of high energy atoms. In the standard methods the update of the neighbor lists is done after a constant number of time-steps. This is a compromise between the need to calculate the neighbor lists every time-step and the computation cost. Our conclusion is that we need a partial update of the neighbor lists. The criterion for updating the neighbor list of an atom may vary from one problem to the other.

For the high energy damage calculation we find that several criteria can be used. We differentiate between two approaches.

In the first approach one checks the high energy atoms of the crystal. In this case we only update the neighbor list every time-step for atoms with

energy higher than a given energy E_{crit}. E_{crit} depends on the time-step size (which is a constant of the problem) and the maximum kinetic energy at the crystal. In order to take into account low energy atoms, and the overall thermal motion in the system, the full neighbor list has to be updated periodically after an appropriate number of time-steps. The second approach is based on the displacement of the atom since the most recent updating of its list of neighbors: if the atom moves more than ΔX in that period, its neighbor list is updated. ΔX is a constant distance, which is determined as a function of the time-step size and the maximum kinetic energy of the atoms in a similar way as for the first approach.

The time needed for updating the neighbor list essentially depends only on the number of high energy atoms and the number of their neighbors. The partial updating of the neighbor list calculation should reduce enormously the CPU time, especially when simulating large systems [6].

4 PARALLELIZING THE ALGORITHMS

The following parallelizations of the algorithms assume many interacting processors. Each processor is an independent computational unit simulating part of the crystal. This corresponds to the MIMD model with shared or distributed memory.

To generate the parallel algorithms it is necessary to overcome three problems. The first involves the distribution of the particles among the different processors in a way that minimizes the cost of data transfer (or the sharing of data) during the computation. The second problem is the modification of the algorithm for the parallel machine. Finally, there is a need to preserve the load balance between the computing processors. For our specific problem, wherever an equal number of atoms is assigned to different processors the computation loads are the same. Since during the simulation (move between processors) process only a few atoms change their locations the computation load is kept in balance in these cases.

4.1 Domain Decomposition

The domain decomposition includes both a simple "particle parallelism" [10] distribution and a special "geometric parallelism" [10] partitioning for the algorithms presented above.

In the particle parallelism method atoms are assigned to processors in an arbitrary way. For this case we define the neighborhood of a computing processor to be those atoms within a distance R_c from its own atoms and which were assigned to other processors. We refer to those processors as the neighboring processors.

In our geometric parallelism the crystal is partitioned into computation cells, where each cell has exactly two neighboring cells. Let V be the volume of the microcrystal defined by the intervals $\langle X_0; X_1\rangle$, $\langle Y_0; Y_1\rangle$, $\langle Z_0; Z_1\rangle$. We slice the volume along $\langle X_0; X_1\rangle$, into P equally sized subvolumes $V_0, \ldots, V_{P-1}$. In our experiments we used P processors numbered from 0 to $P-1$. The atoms in subvolume V_i, $0 \le i \le P-1$ were assigned to the computation cell of processor i.

We view the computation cells as arranged in a torus (V_i and $V_{i+1 \bmod P}$ are neighbors). We define the neighborhood of a computation cell to be the atoms in its neighboring cells which are within distance R_c from the common boundaries (see the overlapping boundaries in Fig. 1). Note that the number of cells needs to be small for small crystals; this number depends on R_c. Thus, one can employ only a small number of processors for simulating small crystals using this method.

4.2 Parallel Implementation of the Algorithms

In this section we modify the two algorithms—relational and refreshment—to run with the particle and geometric distribution methods. We use the following notation:

1. SRMD is the sequential relational algorithm.

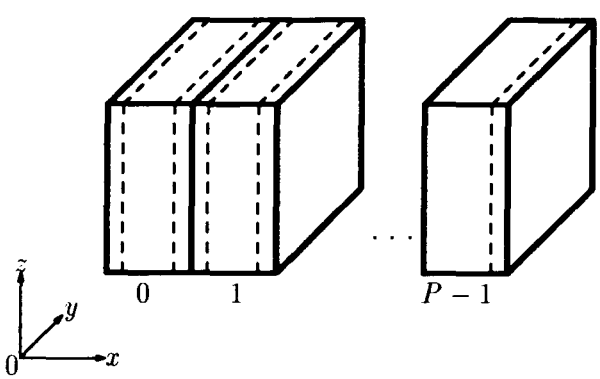

FIGURE 1 Neighboring cells with overlapping boundaries.

2. SFMD is the sequential refreshment algorithm.
3. PRMD is the relational algorithm with particle distribution.
4. GRMD is the relational algorithm with geometric distribution.
5. GFMD is the refreshment algorithm with geometric distribution.

The modifications of the algorithms involve the interaction between the processors. We present this interaction by a stage of messages' exchange. In each of the algorithms, at the end of a time-step, every processor sends all the data about its atoms which are in the neighborhoods of other processors. When a processor receives the data about all the atoms in its neighborhood a new computation cycle begins locally.

In all three parallel algorithms the corresponding procedures were changed to compute subcrystals independently. This allows a concurrent computation of the forces between atoms.

In PRMD the initial distribution of atoms to processors (which is based on the identification numbers of the atoms) remains the same during the entire simulation and each processor is computing the simulation values only for its own atoms. To accomplish this task, each processor receives all the current information on the system. During the communication stage every processor is sending information about the same subset of atoms and receives information from all the processors. Note that algorithmically only information about neighborhoods is needed.

In the geometric parallelism method (the GRMD and GFMD algorithms) the initial distribution of the particles is based on their relative locations. Since atoms may change their relative positions during each time-step, an atom can leave its processor's space and enter the space of a neighbor processor. Thus, the distribution of atoms to processors must be corrected at each time-step. According to our partitioning the atoms can move only between neighboring cells in a single computation cycle and only information from neighboring cells is relevant. Thus, the local memory needed is smaller in comparison to the particle parallelism method but this requires a special modification to the sequential program.

In the particle distribution method, each processor computes locally the global physical characteristics (similar to the sequential algorithm). Unlike this method, in the geometric parallelization we use the "processor farm" approach [3] where a distinct processor collects the local information (from all the processors), computes the global characteristics, and then sends the information to all the other processors. In our parallel program this job is done by the control program ("master" processor).

5 EXPERIMENTAL RESULTS ON THE MEIKO PARALLEL COMPUTER

In this section we report timing results of a few experiments with our algorithms running on a Meiko machine.

Our Meiko machine employs 28 i860 processors with 8 MB or 16 MB of local memory per processor. The communication between the processors is carried out by Transputer (T800) chips (no shared memory). The host computer is a Sun SparcStation which is used for managing the computations on the different processors. The host computer has 28 MB local memory available for computations. The operating system is SunOS 4.1.1 with CSTools 1.19 on top. We use the Portland Group Fortran 77 compiler.

Every value in each experiment is the average of a few trials. The time values are measured in ticks where a tick is counted every 64 microseconds (on the Meiko machine). The logical topology of a system (e.g., a ring topology of computation cells) can be embedded in many physical parts of the Meiko system. Since the physical connections are not known on the Meiko machine, the quality of the communication may depend on the processors selected for the task. Thus, for each topology that was tested a few selections of physical processors were tried out and the best communication method was chosen by experiment.

5.1 Experimental Results for PRMD

The first experiment simulates crystals with 384, 1152, and 5600 atoms for cases with 1, 2, 4, 8, and 16 processors. The temporal performance, $R_T(N; P)$, which is defined as the inverse of the total execution time [8], is shown in Figure 2. In this formulation N is the size of the crystal (number of atoms) and P is the number of processors.

In Figure 3 we compare the communication performance in each of the above cases. We use $R_{com}(N; P)$ to denote the inverse of the total communication time.

From the experiments we learn that when the number of processors is 8, the temporal perfor-

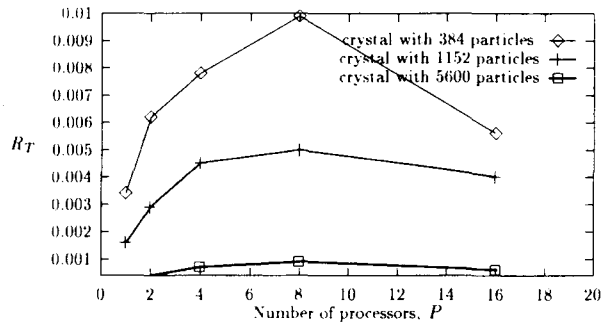

FIGURE 2 Temporal performance. $R_T(N; P)$, for three crystals.

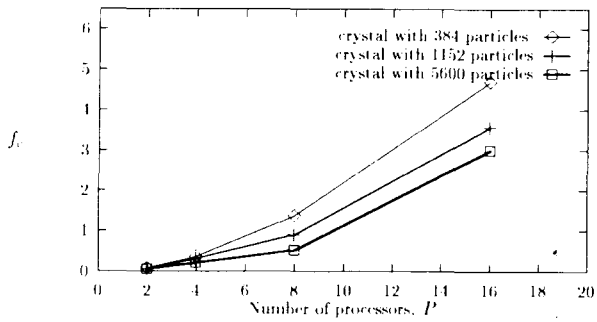

FIGURE 4 The performance overhead $f_c(P)$ for three crystals.

mance is maximized. This can be explained by the communication times as seen in Figure 3. When there are more than 8 processors the total execution time grows in spite of the fact that each processor has a smaller crystal.

Following [5] the relation between the communication and the calculation times is characterized by the performance overhead:

$$f_c(P) = T_{com}(P)/T_{cal}(P)$$

where T_{cal}, T_{com} are the total calculation time and communication time, respectively, spent by the slowest processor.

The performance overhead is compared in Figure 4 for configurations of 2, 4, 8, and 16 processors.

In addition, we conducted several experiments with more than 5,600 atoms (up to 22,400) and four processors. This was the largest crystal that can be computed due to the lack of memory on our units.

5.2 Experimental Results for GRMD and GFMD

To implement this algorithm a master process collects the local information from all the processors and computes the global values (e.g., total energy, etc.). Due to the shortage in memory space on the local processors we use the host computer (SparcStation) to run the master process. Communication with the host is carried out with the XDR package in order to resolve the incompatibility of the numeric representation on the SparcStation and the i860 processors.

We compared the GRMD and GFMD algorithms in Figures 5 and 6 for three crystals.

We can see that the total time gained in GFMD decreases when the number of atoms increases but GFMD is always faster than GRMD. This result is explained by the fact that for more atoms there is a need for more communication. Faster communication channels can improve this phenomenon. Clearly, the geometric parallelization approach is better for large crystals.

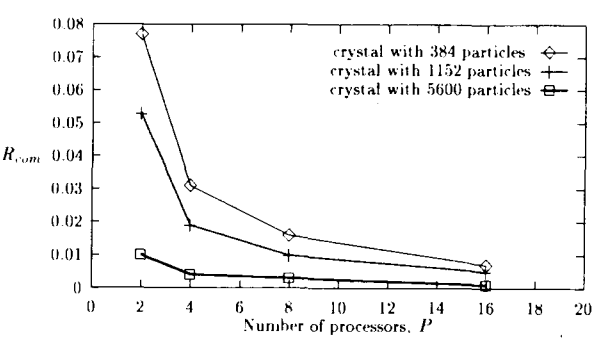

FIGURE 3 Temporal performance of communication time, $R_{com}(N; P)$, for three crystals.

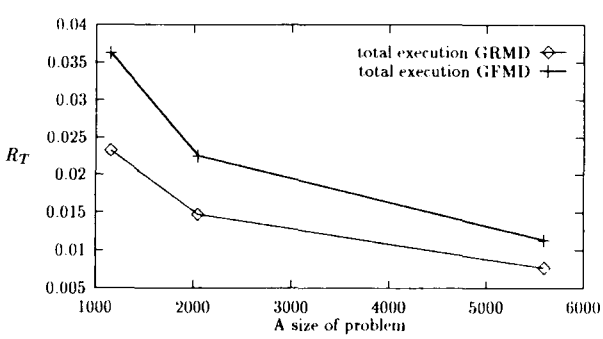

FIGURE 5 Temporal performance of the GRMD and GFMD algorithms for three crystals.

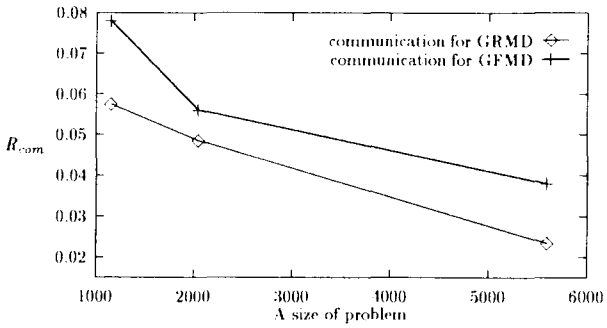

FIGURE 6 Temporal communication performance of the GRMD and GFMD algorithms for three crystals.

5.3 Comparing the Algorithms

In the following experiments we compare the refreshment and the relational algorithms. The five algorithms, each using four processors, simulate 1,152 and 5,600 atoms. The timing results of the most time-consuming procedures are presented in Tables 1 and 2. The results are measured in seconds per time-step.

The particle parallelization method improves on the sequential algorithm by a factor of two. This was achieved in spite of the slow communication channels that consume about 35% to 40% of the total execution time. The geometric parallelization method was supposed to reduce the communication load. This did not happen due to the use of the host computer to run the master process. Nevertheless, this parallelization method improves on the particle parallelization by a factor of two.

The refreshment algorithm improves on the relational in both the parallel and the sequential cases. The refreshment algorithm removes the need to compute the neighbor list in every few time-steps. In the parallel case this improvement almost eliminates the overhead generated by the neighbor list procedure.

6 DISCUSSION AND CONCLUSIONS

We developed new algorithms for simulating high energy irradiation damage using MD calculations. The algorithms were parallelized for a message passing-based architecture and were tested on the Meiko machine. From our experiments we conclude that in every case the parallelization decreases the execution time in spite of the slow communication channels.

The modifications of the molecular dynamic algorithm focused on the computation of the neighbor list. Since this component has a major role in the computation process the improvements were significant.

The success of the geometric parallelization method compared to the particle method can be explained by the reduction in the size of the local subcrystal and that of the communication time.

For future work we plan to graft the principles and ideas applied in the refreshment algorithms into the parallel computation of the interacting forces.

Table 1. Simulation Performance of Five Algorithms on 1,152 Atoms Measured in Seconds Per Time-Step

Algorithm	SRMD	PRMD	GRMD	SFMD	GFMD
Neighbor list procedure	0.325	0.173	0.045	0.095	0.06
Forces procedure	4.68	1.197	0.583	1.233	0.3
Communication	—	0.935	0.435	—	0.32
Total simulation	5.53	2.337	1.08	1.44	0.69

Table 2. Simulation Performance of Five Algorithms on 5,600 Atoms Measured in Seconds Per Time-Step

Algorithm	SRMD	PRMD	GRMD	SFMD	GFMD
Neighbor list procedure	1.075	0.595	0.255	0.223	0.325
Forces procedure	7.81	2.215	1.907	2.01	1.2
Communication	—	1.638	1.07	—	0.65
Total simulation	9.703	4.592	3.285	2.335	2.208

ACKNOWLEDGMENTS

The authors would like to thank the anonymous referees for their insightful reviews. L. Ioffe was supported by the R&D administration of the Ministry of Defense and the ministry of immigration absorption, Israel.

REFERENCES

[1] M. P. Allen and D. J. Teldesley, *Computer Simulation of Liquids*. Oxford: Clarendon Press, 1987.

[2] R. Behrisch, Ed., *Sputtering by Particle Bombardment, I and II, Topics Appl. Phys. Vol. 47*. Heidelberg: Springer Verlag, 1981.

[3] K. Esselink, B. Smit, and P. A. J. Hilbers. "Efficient parallel implementation of molecular dynamics on a toroidal network. Part 1. Parallelizing strategy," *J. Comput. Phys.*, vol. 106, pp. 101–107, 1993.

[4] A. J. E. Foreman, W. J. Phythian, and C. A. English, "The molecular dynamics simulation of irradiation damage cascade in copper using many body potential," *Philos. Mag. A*, vol. 66, pp. 671–695, 1992.

[5] C. G. Fox, M. A. Johnson, G. A. Lyzenga, S. W. Otto, J. K. Salmon, and D. W. Walker. *Solving Problem on Concurrent Processors, General Technique and Regular Problems*, vol. 1. Englewood Cliffs, NJ: Prentice Hall, 1988.

[6] E. Glikman, I. Kelson, N. V. Doan, D. Piccot, G. Pinte, and P. E. Haustein, "Desorption from surfaces induces by neutron capture." *Nuclear Instruments Methods Phys. Res.*, vol. B72, pp. 33–39, 1992.

[7] E. Glikman, I. Kelson, and N. V. Doan. "Molecular dynamics calculations of nuclear stimulated desorption." *J. Vac. Sci. Technol.*, A vol. 9, pp. 2776–2781, Sep./Oct. 1991.

[8] R. W. Hockney, "A framework for benchmark performance analysis," in *Computer Benchmarks*, J. Dongarra and W. Gentzsch, Eds. Elsevier, August 1993.

[9] R. W. Hockney and J. W. Eastwood. *Computer Simulation Using Particles*. New York: McGraw-Hill, 1981.

[10] J. Li, A. Brass, D. J. Ward, and B. Robson. "A study of parallel molecular dynamics algorithms for n-body simulations on a transputer system." *Parallel Comput.*, vol. 14, pp. 211–222, 1990.

[11] V. Rosato, G. Maino, and A. Ventura, "Molecular dynamics calculations for low-energy helium atoms on a nickel surface." *J. Phys. Condensed Matter*, vol. 1, pp. 10021-10037, 1989.

[12] A. Tenenbaum and N. V. Doan, "Point defect migration induced by sub-threshold focused collisions," *Philos. Mag.*, vol. 35, pp. 379–403, 1977.

[13] L. Verlet, Computer experiments on classical fluids. 1. Thermodynamical properties of lennard-jones molecules, *Phys. Rev.*, vol. 159, pp. 98–103, 1967.

[14] H. Zhu and R. S. Averback, An optimization scheme for molecular dynamics simulations of radiation effects. *Nuclear Instruments Methods Phys. Res.*, vol. B83, pp. 334–338, 1993.