УДК 519.6

СРАВНЕНИЕ МЕТОДА ВЕРЛЕТ ТАБЛИЦЫ И МЕТОДА СВЯЗАННЫХ ЯЧЕЕК ДЛЯ ПОСЛЕДОВАТЕЛЬНОЙ, ВЕКТОРИЗОВАННОЙ И МНОГОПОТОЧНОЙ РЕАЛИЗАЦИЙ

Э. С. Фомин[1]

Алгоритмы поиска ближайших соседей широко используются в молекулярной динамике для расчетов короткодействующих межатомных потенциалов. Эти алгоритмы основываются на методах Верлет таблицы и связанных ячеек. Дан анализ особенностей указанных методов и показано, что для плотных систем, таких как вода, метод связанных ячеек требует значительно меньшего объема необходимой памяти и количества операций чтения данных по сравнению с методом Верлет таблицы и может эффективно использоваться в параллельных реализациях. Новая техника для параллелизации расчета короткодействующих потенциалов, названная динамической пространственной декомпозицией, предложена для метода связанных ячеек. Показано, что в параллельной SIMD-версии этот метод превосходит метод Верлет таблицы на 40% и более, несмотря на большое количество излишних расчетов межатомных расстояний. Эффективность обусловлена тем, что данный метод более приспособлен для современных многоядерных SIMD-процессоров. Все методы тестировались на пакете MOLKERN.

Ключевые слова: метод Верлет таблицы, метод связанных ячеек, поиск ближайших соседей, SIMD, многопоточность.

Introduction. The calculation of forces produced by pairwise non-bonded (Coulomb and van der Waals) interactions is a common bottleneck of molecular dynamics programs. These calculations involve the evaluation of the total inter-atomic potential energy V_{total} of N atoms and the forces acting on every atom of the molecular system. Since the total potential energy of the whole system is the sum of pairwise interactions over all atoms, $V_{\text{total}} = \frac{1}{2}\sum V(r_{ij})$, the direct calculation requires $O(N^2)$ steps.

Redundant calculations of inter-atomic potentials can be reduced by using a cutoff distance r_{cut} in potential functions and by assuming that both the potential functions and the forces beyond the cutoff distance are zero. In this way, the number of calculation steps is reduced from $O(N^2)$ to $O(N)$.

The determination of all atom pairs whose atoms are within the cutoff distance is most commonly done by using the improved Verlet table (IVT) algorithm. This approach combines the following two methods: the conventional Verlet table (VT) algorithm and the cell-linked list (CLL) algorithm. On the average, for a low density system, such as the gas state of argon [2], the IVT outperforms both the VT and the CLL algorithms by a factor of more than two in both serial and parallel programs. This method is used in most modern molecular dynamics programs.

Nevertheless, a comprehensive analysis of the algorithms under discussion and of the ways in which they can be implemented in parallel calculations shows that the point is somewhat questionable. There are some differences to consider, since these methods are differently applicable on parallel platforms, and different degrees of efficiency can be achieved.

1. Neighbor search algorithms.

1.1. Cell-linked list method. In this approach [1] the simulation domain is partitioned into cubic cells with edges equal at least to r_{cut}, and every atom is assigned to one of these cells according to their coordinates. The neighboring atoms of an atom can be listed by enumerating all atoms in all the 26 adjacent cells and the cell under consideration itself. Since the adjacent cells of each cell are known and they will not change during the simulation, the cell neighbor list table is usually constructed at the beginning of the simulation. The assignment of each atom to the cells is updated before each potential and force evaluation step.

It should be noted that the operation of assigning an atom to the cells has the constant $O(1)$ complexity. For example, the assignment can be defined as $\{n_x, n_y, n_z\} = \left\{[x/r_{\text{cut}}], [y/r_{\text{cut}}], [z/r_{\text{cut}}]\right\}$, where $\{n_x, n_y, n_z\}$ is the three-dimensional index of a cell, $\{x, y, z\}$ are the atom coordinates, and $[x]$ is the floor function, i.e., the

[1] Институт цитологии и генетики СО РАН, просп. акад. Лаврентьева, 10, 630090, г. Новосибирск; ст. науч. сотр., e-mail: fomin@bionet.nsc.ru

largest integer less than or equal to x. Thus, it takes $O(N)$ steps to update the atom lists for all cells. So, an efficient atom list update is one of the advantages of the CLL method. The total number of pairs that can be constructed for an atom by using all neighboring atoms is equal to $27\rho r_{\text{cut}}^3$, where ρ is the atom density, and only $(4\pi/3)\rho r_{\text{cut}}^3$ of them have atoms within the cutoff distance. The CLL method, which generates a number of pairs about $81/4\pi = 6.45$ times greater than necessary, therefore performs a large number of unnecessary interparticle distance calculations.

1.2. Verlet table method. The idea of the Verlet table (VT) method [1] is to construct and maintain a list of neighboring atoms for every atom in the system. In this method, the cutoff sphere of radius r_{cut} is surrounded by a larger "skin" sphere of radius r_{skin}. An atom is considered to be a "neighbor" if the distance between the two atoms is shorter than r_{skin}. Each atom is assumed to interact only with those in its neighbor list, and this neighbor list should be updated periodically for a fixed interval or reconstructed automatically whenever some atoms move too much and the list will be out-of-date.

In the conventional VT method, the construction/update of the neighbor table requires $O(N^2)$ steps of interatomic distance evaluation, which is the major drawback of the method. Another shortcoming is that, as the number of atoms and/or r_{skin} increases, the memory demand for the neighbor table becomes unreasonable. Every row of the table requires $(4\pi/3)\rho r_{\text{skin}}^3$ elements, and for a typical calculation with $\rho = 0.1$ Å^{-3} and $r_{\text{skin}} = 10$ Å this value approaches 500 elements. Due to the storage of all atoms inside a larger "skin" sphere instead of the cutoff sphere, the Verlet table contains about $(r_{\text{skin}}/r_{\text{cut}})^3$ times more pairs than necessary. For calculation with $r_{\text{skin}}/r_{\text{cut}} = 1.1$, this table contains slightly more than 75.1% of the particle pairs for which pairwise distances are within r_{cut}.

The VT method has been proven to be efficient when the atoms of the entire system move slowly. As the atoms move more quickly, either the "skin" radius or the frequency of reconstructing the Verlet table should be increased. Both of these requirements cause a dramatic increase in the CPU time used to maintain the Verlet table, and the whole simulation becomes inefficient.

1.3. Improved Verlet table method. The improved Verlet table method (IVT) [2] combines the advantages of the VT and the CLL. Like the CLL method, the whole simulation domain is partitioned into cubic cells with edges equal to r_{skin} and all atoms are assigned to the corresponding cells according to their coordinates. Then the neighbor list table is constructed, but only atoms in the neighbor cells rather than all the atoms in the system are considered in the evaluation of interatomic distances. Since the IVT method eliminates the major drawback of the VT method, i.e., the construction of the table in $O(N^2)$ steps, its efficiency is very high in comparison with the conventional VT method. However, the second shortcoming of the VT method, the high memory demand, still remains.

1.4. Improved cell-linked list method. The cell-linked list method improved by P. Gonnet [3] and hereafter referred as the improved cell-linked list method (ICLL) reduces the number of unnecessary interparticle distance calculations by sorting particles along the cell pair axis and by selecting two particles if their distance along the axis is smaller than the cutoff distance. For sorting particles along the cell pair axis, the quicksort algorithm is used, which scales as $O(N \log N)$. According to [3], the neighbor list of the ICLL method contains $\approx 59.4\%$ of the particle pairs for which the pairwise distance is within r_{cut} vs. $\approx 16\%$ in the CLL method, i.e., four times better. The ICLL method is better than other methods [4] that employ the partitioning of the domain into cells of edge lengths smaller than r_{cut}. The number of spurious distance pairs in the IVT and ICLL methods is almost the same for typical calculations, which means that they are, in this respect, equally efficient.

2. Implementation of the methods.

2.1. Algorithms of feature analysis. Usually, the efficiency of serial and parallel implementations are influenced by both the choice of the algorithms and by the processor features. The more features of the target processor architecture an algorithm takes into account, the higher efficiency can be achieved. The best performance is achieved when the algorithm is flexible enough to be most closely adapted to the target processor architecture.

For both Verlet table algorithms, it is natural to choose an integer holding the atom index as an element of the Verlet table. We have no other choice due to the memory requirements for the table. As mentioned above, the Verlet table has to store $(4\pi/3)\rho r_{\text{skin}}^3 N$ elements, and under typical simulation conditions the factor $(4\pi/3)\rho r_{\text{skin}}^3$ is up to 500 entries. In the case of simulation of a large system containing about 10^6 atoms, therefore, the Verlet table must hold nothing but integers. The shortcoming of such index storing appears at the time of data transfer between memory and processing units. Due to the fact that a Verlet table stores nonsequential atom indices, the memory access to the atoms has to be performed one by one and the total number of atom data transfer operations has the same order of magnitude as the table size, i.e., $(4\pi/3)\rho r_{\text{skin}}^3 N$.

For cell-linked list methods, the memory requirement to the number of elements of cell atom lists is N,

i.e., the number of atoms of the whole system. Thus, it allows any moderate data structures but not only array indices to be stored in cell atom lists. Also, the total number of data transfer operations between memory and processing units can be estimated as $27\rho r_{\text{cut}}^3 N_{\text{cell}}$, where N_{cell} is the number of cells, or as $27N$, because $\rho r_{\text{cut}}^3 N_{\text{cell}}$ is equal to N. The factor 27 appears due to the fact that every cell should be used in calculations alone and in combination with all adjacent cells. As a result, the CLL methods reduce both memory size and the number of data transfer operations significantly. The gain can be as large as $\big(4\pi/3\big)\rho r_{\text{skin}}^3$ in memory size and $\big(4\pi/3\big)\rho r_{\text{skin}}^3/27$ in transfer operations. The freedom in choosing the table element type is also essential. It allows atom copies to be stored in cell atom lists instead of array indices. The elements of these atom copies can be formatted as 4-element arrays and aligned to 16-byte memory boundaries to be efficiently used by Streaming SIMD Extensions (SSE). Also, they can be accessed through direct memory access (DMA) instructions not only individually but also in groups. The last possibility is essential for specialized high-performance processor architectures, such as the Cell Broadband Engine Architecture (CBEA) [5] and General-Purpose Graphics Processing Units (GP GPU) [6].

The second drawback of the Verlet table is the asymmetrical "master–slave" atom pair storage. The row index refers to a "master" atom and the row elements indicate all "slave" atoms, i.e., the atoms located within the cutoff distance of the "master" atom. According to the usual practice, the table stores no doublets, i.e., the nth row saves atom indices with values $k > n$ and ignores the indices with values $k < n$. In this way, the table size is reduced by a factor of two and the uniqueness of each atom pair is ensured. Thus, the full energy can be efficiently calculated as the total sum over the pairs in either serial or parallel programs without any inconvenience. However, the total force cannot be calculated in such a simple way. For instance, we can write the force acting on the nth atom as $F_n = \sum\limits_{k<n} F_{nk} + \sum\limits_{k>n} F_{nk}$, where F_{nk} is the force acting on the nth atom by the kth atom. Due to the fact that the nth row of the table contains the atom indices with $k > n$, the second sum can easily be calculated and the result can be sent to the computer memory cell accumulating F_n. However, all terms in the first sum should be calculated and sent to F_n individually. That can be done efficiently in serial programs but not in parallel programs due to potential memory contentions. The Verlet table enlargement does not remove all difficulties. The insertion of all atoms located within the cutoff distance of a "master" atom into the "master" row allows an efficient summation of the forces but doubles the amount of computation.

The cell-linked list methods store atoms without any asymmetry in the cell atoms lists. It is sufficiently easy to share calculations between threads without memory contentions. A technique by which it can be done is proposed below. It is named the dynamic spatial decomposition and will be explained in the following section.

Thus, the shortcomings of the Verlet table methods inevitably complicate the parallel implementation and lower the efficiency. By contrast, the cell-linked list methods are free of these shortcomings and allow efficient parallel implementations.

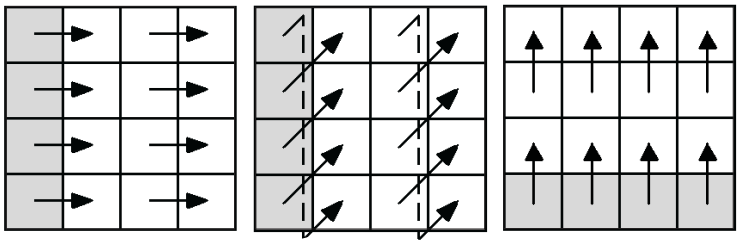
2.2. Dynamic spatial decomposition. The efficiency of any parallel implementation is greatly dependent on the technique used. The most commonly used techniques for parallelizing short-range interactions in molecular systems are known as the replicated data (RD) [7], spatial decomposition (SD), and index decomposition (ID) methods [8]. In the RD method, each running thread stores a copy of all atom data needed in the simulation. This technique allows avoiding memory contentions by creating many replicas of output data, one per running thread, and merging all replicas after every calculation step. The drawback of RD is that its memory and communication cost scales as $N * K$, where N is the number of atoms and K is the number of running threads. Thus, this technique is efficient only for few running threads. In the SD technique, the simulation domain is broken into pieces, one per running thread. Each running thread computes the forces acting on only the atoms in its subdomain, thereby avoiding memory contentions. The SD technique scales optimally as N/K, since each running thread demands only information from threads that own neighboring subdomains. The shortcoming of this technique is that the calculation burden doubles for all the pairs whose atoms are assigned to adjacent subdomains. In the ID technique, the atom array is broken into pieces of about equal sizes, with every piece being assigned to a running thread. This technique has the same shortcomings as the SD, but it is more convenient for code implementation.

In the implementation of the CLL methods, we propose a new technique referred to as the dynamic spatial decomposition. It avoids the shortcomings of both the replicated data and spatial/index decomposition techniques. In this technique, all the spatial cells of the simulation domain are grouped in couples. An example is shown in the figure, part a. All short-range interactions between atoms are calculated only for coupled adjacent cells, and the results are accumulated. At the next step, the spatial cells are regrouped into other couples, for example, as shown in the figure, part b, and the calculation is repeated. Then the spatial cells are regrouped again, for example, as shown in the figure, part c, and so on. It takes at least 26 such steps to calculate all the

short-range interactions between the atoms of all coupled adjacent cells according to the number of neighbors of any spatial cell. The calculation of the interactions inside the spatial cells themselves is carried out at the final 27th step.

Any two cell couples are independent of each other with regard to the data, and every cell couple and/or the spatial cell itself can be sent to a separate thread of execution. As for the computation of short range interactions, the dynamic spatial decomposition scheme affords the use of up to $N_{\text{cell}}/2$ simultaneously running threads, where N_{cell} is the number of spatial cells of the simulation domain. For example, a simulation of water in a $320 \times 320 \times 320$ Å^3 box, which contains 1.07 million water molecules, can employ up to 16384 threads.

An efficient well-balanced parallel implementation code can be obtained under the next two conditions. First, the simulation domain should be divided into couples without a remainder, which is possible, for example, in the case of even numbers of cells in every dimension. Second, the number of threads should be a divisor of $N_{\text{cell}}/2$ to maintain equal numbers of coupled cells per running thread. With these two conditions, a code can be written in a way that would minimize the delays resulting from thread synchronization.



Different examples of cell coupling

2.3. Implementation and tests. All the IVT, CLL, and ICLL methods were implemented in the framework of the MOLKERN simulation software [9]. The following three implementations were developed for each method:

1) serial C++ code,
2) vectorized C++ code,
3) multithreaded vectorized C++ code.

The vectorized codes were written by using the SSE and SSE2 intrinsics. The multithreaded versions were constructed with the help of the boost thread library [10].

In the IVT method, the Verlet table stores the atom identifiers only due to the shortcomings mentioned above. For the sake of memory saving, the Verlet table stores the atom pairs without any doublets. In the multithreaded versions, SD was used for the energy and forces accumulation and RD to avoid memory contentions during resulting force storage. The latter demands only $K * O(N)$ extra steps for merging the resulting forces, where K is number of threads used. So, it is very efficient for small numbers of threads, the time overhead being no more than 1% of the calculation time.

As for the CLL and ICLL methods, the spatial cells of the simulation domain store the atom identifiers as well as the copies of all data required for calculations. Additionally, the data of the CLL were formatted as 4-element arrays and aligned to 16 byte memory boundaries to be efficiently used by the SSE instructions. The dynamic spatial decomposition scheme was used for parallelization.

We used simulations consisting of 2.1, 16.8, 134.1 thousand and 1.07 million SPC water molecules in an $L \times L \times L$ Å^3 box, where $L = \{40, 80, 160, 320\}$ Å, respectively, at 300 K with all interactions truncated at 10 Å. The Coulomb $1/r$, Lennard–Jones $(\varepsilon/4)\left[(\sigma/r)^{12} - (\sigma/r)^{6}\right]$, and the short-range Coulomb $\operatorname{erfc}(\sqrt{\pi}\, r/r_{\text{cut}})/r$ potentials were employed. The performance is measured in $(\text{atoms} * \text{step} \,/\, \text{second})$ and its value can be calculated as the product of the number of atoms by the number of molecular dynamic steps divided by the whole simulation time, i.e., the number of atoms per second. The time required for the calculation of the valent O–H bonds and the valent H–O–H angles was excluded. All simulations were run on a single node of the HP ProLiant BL2x220c G5 blade server [11], where each blade node includes two Intel Quad-Core Xeon E5450, 3 GHz processors.

3. Numerical results.

3.1. Serial version. The performances of different algorithms for molecular dynamics simulation are compared in Table 1 (the Coulomb $1/r$ potential) and Table 2 (the short-range Coulomb $\operatorname{erfc}(\sqrt{\pi}\, r/r_{\text{cut}})/r$ potential). The simulations were done for SPC water molecules in an $L \times L \times L$ box with $L = \{40, 80, 160, 320\}$ Å and with all interactions truncated at 10 Å. We compared the IVT, CLL, and ICLL methods. The last two columns show the ratios of performances of the cell-linked list methods and the improved Verlet table.

To compare the results, we wanted to demonstrate how the aforementioned features of methods affect the efficiency of these methods. For this purpose, we set preferences for the IVT. First, we eliminated all spurious

Table 1

Comparison of three algorithms for spatial domains of different sizes in the serial running mode. The Coulomb $1/r$ potential is used

Spatial domain, $\mathring{A}^3$	IVT	CLL	ICLL	CLL/IVT	ICLL/IVT
$40 \times 40 \times 40$	63690	35382	57900	0.55	0.91
$80 \times 80 \times 80$	87225	48645	79050	0.56	0.91
$160 \times 160 \times 160$	84885	47394	78432	0.56	0.92
$320 \times 320 \times 320$	85161	47634	78573	0.56	0.92

Table 2

Comparison of three algorithms for spatial domains of different sizes in the serial running mode. The short-range Coulomb $\operatorname{erfc}\left(\sqrt{\pi}\, r/r_{\text{cut}}\right)/r$ potential is used

Spatial domain, $\mathring{A}^3$	IVT	CLL	ICLL	CLL/IVT	ICLL/IVT
$40 \times 40 \times 40$	15165	12999	14811	0.86	0.99
$80 \times 80 \times 80$	15567	13176	15102	0.85	0.97
$160 \times 160 \times 160$	15534	13191	14958	0.85	0.96
$320 \times 320 \times 320$	14226	13284	14865	0.93	1.04

distance pairs from the table by assuming the skin radius to be equal to the cutoff radius and transferring the pair sifting code to the Verlet table update step. Second, we excluded the time for the neighbor atom table creation/update from the results at all. With both these assumptions, the best performance of the method is achieved, which cannot be reached in actual calculations. Without them, the IVT method may lose more than 10% of its performance.

No such preferences were done for the CLL methods. Both CLL methods demand atom pair sifting, and the time of this process was included into the results. It should to be mentioned that the pair sifting process includes the following two steps:

1) removal of all the pairs with distances between the atoms exceeding the cutoff radius,

2) removal of the pairs whose atoms belong to the same molecule and are chemically bonded to each other.

The data presented in Table 1 indicate that in the case of a simple potential, such as Coulomb $1/r$, the CLL method is significantly, about twofold, less efficient than the IVT. As to the more complicated short-range Coulomb $\operatorname{erfc}\left(\sqrt{\pi}\, r/r_{\text{cut}}\right)/r$ potential, the speed-down of the CLL methods in comparison with the IVT is less, no more than 15%. This can be explained by the change in the relative fraction of the potential calculation time with regard to the time of spurious distance pair sifting.

The performance of the ICLL method in comparison with the IVT method is insignificantly worse, the loss being within 10%. So, with regard to the conditions of our tests, it means that the IVT and ICLL methods are almost equally efficient in real calculations for serial runs.

3.2. Vectorized version. The comparison of algorithm performances for the vectorized versions is shown in Table 3. The Coulomb $1/r$ and Lennard–Jones $(\varepsilon/4)\left[(\sigma/r)^{12} - (\sigma/r)^{6}\right]$ potentials were used in combination. Simulations were done for SPC water molecules under the same conditions as used above.

Table 3

Comparison of three algorithms for spatial domains of different sizes in the vectorized running mode. The Coulomb $1/r$ and Lennard–Jones $(\varepsilon/4)\left[(\sigma/r)^{12} - (\sigma/r)^{6}\right]$ potentials are used in combination

Spatial domain, $\mathring{A}^3$	IVT	CLL	ICLL	CLL/IVT	ICLL/IVT
$40 \times 40 \times 40$	142164	149508	204135	1.05	1.44
$80 \times 80 \times 80$	126480	133137	175059	1.05	1.38
$160 \times 160 \times 160$	130638	137700	166956	1.05	1.28
$320 \times 320 \times 320$	130299	135582	166005	1.04	1.27

Table 3 shows that the performance of the CLL method is slightly better than that of the IVT method for vectorized implementations and the performance of the ICLL method is much better, by about 30%. The main cause of the fact that the IVT method is inferior to the CLL and ICLL methods, as follows from the analysis of different modifications of the code, is connected to its "master–slave" data storage, which causes some difficulties in the aforementioned accumulation of the forces. The exclusion of force accumulation was found to decrease the ratio of performance to 0.95 in the case of CLL/IVT and to 1.2 in the case of ICLL/IVT.

3.3. Multithreaded vectorized version. The comparison of performances of multithreaded vectorized versions of the algorithms is presented in Table 4 and Table 5. The results are shown as functions of the number of threads used. Simulations were done for SPC water molecules (Table 4) and for uniformly distributed argon atoms with the same density (Table 5) in a $320 \times 320 \times 320$ Å^3 box. The Coulomb $1/r$ and Lennard–Jones $(\varepsilon/4)\Big[(\sigma/r)^{12} - (\sigma/r)^6\Big]$ potentials were used in combination for SPC water molecules and only the Lennard–Jones $(\varepsilon/4)\Big[(\sigma/r)^{12} - (\sigma/r)^6\Big]$ potential for argon atoms. All interactions were truncated at 10 Å.

Table 4

Comparison of three algorithms for different numbers of threads in the multithreaded vectorized running mode. Simulations were done for SPC water molecules in a $320 \times 320 \times 320$ Å^3 box. The Coulomb $1/r$ and Lennard–Jones $(\varepsilon/4)\Big[(\sigma/r)^{12} - (\sigma/r)^6\Big]$ potentials are used in combination

threads	IVT	CLL	ICLL	CLL/IVT	ICLL/IVT
2	253596	256434	328614	1.01	1.29
4	477048	493170	602352	1.03	1.26
8	812796	884448	1084644	1.08	1.33

Table 5

Comparison of three algorithms for different numbers of threads in the multithreaded vectorized running mode. Simulations were done for argon atoms randomly and uniformly distributed over a $320 \times 320 \times 320$ Å^3 box. The Lennard–Jones $(\varepsilon/4)\Big[(\sigma/r)^{12} - (\sigma/r)^6\Big]$ potential is used.

threads	IVT	CLL	ICLL	CLL/IVT	ICLL/IVT
2	268440	326276	379768	1.21	1.41
4	502748	574956	690590	1.14	1.37
8	868420	1074482	1177320	1.23	1.36

Here is the main difference between the two cases. In the first case (Table 4), there are many pairs whose atoms are bonded. They include all pairs whose O and H atoms belong to the same water molecules. All these pairs must be selected to avoid calculations of the Coulomb and Lennard–Jones potentials between them. In the second case (Table 5), no pair has chemically bonded atoms. Therefore, the sifting of bonded atoms is avoided and the efficiency of calculations increases.

The simulation results once more confirm the superiority of the CLL methods over the IVT. As can be seen, the CLL outperforms the IVT by up to 10% for calculations of molecules and by up to 20% for calculations of chemically nonbonded atoms. The ICLL method outperforms the IVT by one-fourth and one-third, respectively.

4. Conclusions. We present a comparison of the efficiencies of a set of top methods, such as the improved Verlet table, the cell-linked list and improved cell-linked list approaches employed for substantial speed-up of molecular dynamics simulations with short-ranged pairwise potentials. We have analyzed the features of the methods and have found that the CLL methods reduce both the memory size and the number of data transfer operations significantly in comparison with the VT methods. The gain can be as large as $(4\pi/3)\rho r_{\text{skin}}^3$ in memory size and $(4\pi/3)\rho r_{\text{skin}}^3/27$ in data transfer operations. We have made an implementation of the CLL methods that utilizes this gain and have achieved a very good efficiency for parallel running. A new technique for parallelizing short-range interaction calculations, the dynamic spatial decomposition, is proposed for the CLL approach. For

this technique, an efficient well-balanced parallel implementation code can be obtained under the conditions of even numbers of cells in every dimension and of thread numbers being divisors of $N_{\text{cell}}/2$, where N_{cell} is the number of cells in the simulation domain.

We have implemented the IVT, CLL, and ICLL methods in the MOLKERN simulation software by using the same approaches for data storage, extraction and processing. We also used the same procedures for calculations of short-range interactions. Thus, an appropriate comparison of different methods can be performed. We made the efficiency measurements under conditions that are preferential for the IVT but not for the CLL. It has been shown that, even with these preferences for the IVT, the CLL method, especially as improved by P. Gonnet, outperforms the improved VT method by up to 40% or more in spite of a large number of unnecessary inter-particle distance calculations. Thus, it is expected that the CLL methods will outperform the IVT in actual calculations even more.

5. Acknowledgements. This work was supported by the Interdisciplinary SB RAS Basic Research Integration projects N 26 "Mathematical models, numerical methods, and parallel algorithms for large tasks from the Siberian Branch of the Russian Academy of Sciences and their implementation on multiprocessor supercomputers", N 113 "Development of computation methods, algorithms, hardware, and software for parallel simulation of natural processes", N 119 "Post-genomic bioinformatics: computer analysis and modeling of molecular-genetic systems" and by the RAS program N 22.8 "Molecular and cellular biology". This work was also partially supported by the Russian Ministry of Education and Sciences (contract N P857).

СПИСОК ЛИТЕРАТУРЫ

1. *Allen M.P., Tildesley D.J.* Computer simulation of liquids. New York: Oxford University Press, 1990.
2. *Yao Z., Wang J.-S., Liu G.-R., Cheng M.* Improved neighbor list algorithm in molecular simulations using cell decomposition and data sorting method // Comput. Phys. Commun. 2004. **161**. 27–36.
3. *Gonnet P.* A simple algorithm to accelerate the computation of non-bonded interactions in cell-based molecular dynamics simulations // J. Comput. Chem. 2007. **28**. 570–573.
4. *Mattson W., Rice B.M.* Near-neighbor calculations using a modified cell-linked list method // Comput. Phys. Commun. 1999. **119**. 135–148.
5. *Kahle J.A., Day M.N., Hofstee H.P., Johns C.R., Maeurer T.R., Shippy D.* Introduction to the Cell multiprocessor // IBM J. of Research and Development. 2005. **49**, N 4/5. 589–604.
6. *Anderson J.A., Lorenz C.D., Travesset A.* General purpose molecular dynamics simulations fully implemented on graphics processing units // J. Comput. Science. 2008. **227**. 5342–5359.
7. *Smith W.* Molecular dynamics on hypercube parallel computers // Comput. Phys. Commun. 1991. **62**. 229–248.
8. *Fincham D.* Parallel computers and molecular simulation // Molecular Simulation. 1987. **1**. 1–45.
9. *Fomin E.S., Alemasov N.A., Chirtsov A.S., Fomin A.E.* MOLKERN: A library of software components for molecular modeling programs // Biophysics. 2006. **51**, Suppl. 1. 110–112.
10. http://www.boost.org
11. http://www2.sscc.ru

Поступила в редакцию
16.07.2010