Improved $O(N)$ neighbor list method using domain decomposition and data sorting

Zhenhua Yao
The Singapore-MIT Alliance,
National University of Singapore,
Singapore 117576

Jian-Sheng Wang
Department of Computational Science,
National University of Singapore,
Singapore 119260

Min Cheng
Department of Communication Engineering,
Nanyang Technological University,
Singapore 639798

***Abstract*— The conventional Verlet table neighbor list algorithm is improved to reduce the number of unnecessary interatomic distance calculations in molecular simulations involving large amount of atoms. Both of the serial and parallelized performance of molecular dynamics simulation are evaluated using the new algorithm and compared with those using the conventional Verlet table and cell-linked list algorithm. Results show that the new algorithm significantly improved the performance of molecular dynamics simulation compared with conventional neighbor list maintaining and utilizing algorithms in serial programs as well as parallelized programs.**

I. INTRODUCTION

Some molecular simulation techniques such as molecular dynamics and Monte Carlo method are widely used to study the physical properties and chemical processes which contain large amount of particles at the atomic level in statistical physics, computational chemistry, molecular biology field [1]. All these methods involve evaluation of the sum of total interatomic potential energy V_{tot} of N atoms and/or the gradients of potential energy. The interatomic potential contains various interactions between atoms in the physical system, and are usually functions of internal coordinates of atoms, or can be expressed by interatomic distances between two atoms, bond angles among three atoms, etc. For example in molecular dynamics, the potential energy of liquids and gases are often described as a sum of two-body (or pairwise) interactions over all atom pairs. A common practice of two-body interatomic interaction is expressed by Lennard-Jones potential function, which is a simple function of the distance r_{ij} between atom i and j and in shown as follows,

$$V_{\text{LJ}}(r_{ij}) = \frac{\epsilon}{4}\left[\left(\frac{\sigma}{r_{ij}}\right)^{12} - \left(\frac{\sigma}{r_{ij}}\right)^{6}\right] \tag{1}$$

and the total potential energy of whole system is the sum of pairwise interactions over all atoms,

$$V_{\text{tot}} = \frac{1}{2}\sum_{i=1}^{N}\sum_{\substack{j=1\\ j\neq i}}^{N} V_{\text{LJ}}(r_{ij}) \tag{2}$$

In molecular dynamics simulation, evaluation of Eq.(2) and its gradients costs usually most of CPU time. Apparently direct calculation of Eq.(2) requires N^2 steps. If we notice that $V(r_{ij}) = V(r_{ji})$, the total calculation steps can be decreased to $N(N-1)/2$. Obviously it is impactible to carry out such a heavy calculation when the number of atoms is large, and some methods to reduce the redundant computation related to evaluation of Eq.(2) are strongly needed.

General way to reduce the number of calculation of Eq.(2) is using a cutoff distance r_{cut} in potential functions, and assume that both potential functions and its gradients beyond the cutoff distance are zeroes. This treatment reduces the computing time greatly by neglecting all atoms beyond the cutoff distance, since interactions between these atoms are zeroes and needn't to be considered. A straightforward way to determine which atoms are within cutoff distance is to evaluate all distances over all atoms pairs, and this procedure needs $N(N-1)/2$ steps.

A reduction of redundant calculation of interatomic potential can be accomplished by conventional Verlet table algorithm and cell-linked list algorithm.

The basic idea of Verlet table method is to construct and maintain a list of neighboring atoms for every atoms in the system. During the simulation, this neighbor list will be updated periodically for a fixed interval or reconstruct itself automatically when some atoms move too much and the list is going to be out-of-date. During the interval of neighbor list updating, each atom is assumed to interact only with those in its neighbor list. Of course constructing of the Verlet table needs to carry out $N(N-1)/2$ times of interatomic distance evaluation. The Verlet table method has been proven to be efficient when a system contains a relatively small number of atoms and the atoms move slowly. However, the main drawback is that as the number of atoms increases, the memory requirement for maintaining the neighbor lists becomes forbidable, and the time to construct a Verlet table increases as the order $O(N^2)$. Moreover, as the atoms move more quickly, either the "skin" (largest distance allowable of an atom that won't make the neighbor list invalid) must increase or the frequency of reconstructing Verlet table must increase. Both of these requirements make CPU time used to maintain the Verlet table increase dramatically, and the whole simulation becomes inefficient.

Cell-linked list algorithm is another effective method to reduce the calculation of potential and force evaluation when the number of atoms is large. This approach partitions the simulation domain into small cells, and every atoms are assigned to these cells by their coordinates. Since the neighborhood cells of each cells are known and won't change during the simulation, and the lengths of three edges (for three dimensional simulation domain) of cells can be selected to equal to r_{cut}, then neighborhood atoms of an atom can be listed by enumerating all atoms in all neighborhood cells and the cell itself. The implementation of cell-linked list algorithm is usually to construct a cell neighbor list table at first, and then assign each atoms to the cells before potential and force evaluation step. The first process requires little CPU time and is need to be carried out only once, and the second process needs the order of $O(N)$ steps to be carried out. The overhead to update the neighbor list, i.e., assign each atom to corresponding cells, is very small, but considering there are 26 surrounding cells for each cells, a big number of interatomic distance still need to be calculated, in which usually over 80% are fall outside the cutoff distance, and this makes cell-linked list method rather inefficient compared to Verlet table method when the number of atoms is small.

Basically there is a tradeoff between overhead for maintaining neighbor list table and reduction of calculation of unnecessary interatomic distance.

In this paper conventional Verlet table method is improved and the overhead to maintain the neighbor list table has been reduced to the order of $O(N)$, and the efficiency of calculating interatomic distance is still as high as of Verlet table and cell-linked list method in almost all instances. Furthermore it is easy to parallelize on SMP platforms as well as on workstation clusters.

Conventional Verlet table algorithm and cell-linked list algorithm have been widely parallelized and have shown significant reduction in total computing time. In this work it is intend to optimize serial performance on single processor as well as parallel environment. The modifications of the algorithm and demonstration of the improved performance on single processor computer and dual–processors are described in this paper. Moreover the molecular dynamics program based on the improved Verlet table method has shown good scalability on Linux workstation cluster system.

II. Method of improved Verlet table algorithm

The conventional Verlet table method and cell-linked list method have been detailed in a classical book about molecular simulations by Allen and Tildesley [5]. So the implementation details of basic algorithms are not written down in this paper.

A. Conventional Verlet table algorithm

In conventional Verlet table algorithm the potential cutoff sphere of radius r_{cut} is surrounded by a "skin", to give a larger sphere of radius r_s [5]. At the first step of simulation a neighbor list is constructed for every atoms in the system, and an atom is considered as a "neighbor" if the distance between two atoms is equal to or shorter than r_s. Over the next few time steps this neighbor list is used in the force and potential evaluation routine. For every atoms, only interatomic distance of which atoms in its neighbor list table are calculated, all other atoms are skipped, thus a huge amount of unnecessary interatomic distance calculation is eliminated and the overall performance is increased. In the following from time to time, the neighbor list is reconstructed and the similar procedure is repeated. It should be noticed that the "skin" around the cutoff distance is chosen to be large enough so that between reconstruction intervals any atoms which is not in the neighbor list of an atom cannot penetrate through the skin into the cutoff sphere.

The interval between updating of neighbor list table can be a fixed value in the program, and this value varies for different system with different atom mobility. Neighbor list can also be automatically updated by monitoring the accumulation of atoms' displacement vectors, when difference of any two atoms' displacement vectors is large enough, the neighbor list is reconstructed.

In conventional Verlet table algorithm, it is need to evaluate the interatomic distances between all atom pairs, so the total steps to construct a neighbor list table is the order of $O(N^2)$. But once the neighbor list is constructed and between the interval of updating, evaluation the forces / potentials of the system is efficient because there are only atoms in the neighbor list, i.e., in the sphere of r_s as the radius, need to be evaluate the interatomic distances, and this procedure requires the order of $O(N \cdot N_{\text{neighbor}})$ steps, in which N_{neighbor} is the average number of neighbors in the material and won't change with the system size.

The advantage of conventional is the efficiency of using neighbor list in the evaluation of forces / potentials, as the average number of neighbors in the list is only a few more than actual needed (they are atoms which fall inside the "skin"). Its disadvantage is the inefficiency of constructing neighbor list, as the procedure requires the order of $O(N^2)$ steps (more precisely, $N(N-1)/2$ steps). Therefore if the reconstruction algorithm can be improved, it will be a good choice in molecular simulation.

B. Cell-linked list algorithm

In the conventional cell-linked list algorithm the simulation space is partitioned into several cells, and each edges of cells are equal to or larger than cutoff distance of the potential function. All atoms are assigned to the cells according to their positions, and during this procedure a linked list of the atom indices is created. At the beginning of a simulation, an array that contains a list of cell neighbors for each cells is created, and this list remains fixed unless the simulation domain changes during the simulation [5].

Any cell should be one of the neighbors of a cell if it has at least one point within the cutoff distance of any point. Since in the conventional method all edges of each cell are equal to or larger than r_{cut}, considering the periodic boundary condition any cell has 8 neighbors (for two dimensional domain) or 26 neighbors (for three dimensional domain). All atoms not located in neighbors of a cell are fall outside of potential function cutoff distance.

A modification to cell-linked list algorithm is to reduce the cell size so that the possible neighbors of atoms can be reduced, as described in Ref [5].

In the conventional method the edges of cells are often chosen to the cutoff distance r_{cut}, then for any cells all atoms in 27 cells, or in the volume of $27 \times r_{\text{cut}}^3$, will be evaluate the interatomic distances. Ideally, only atoms in the volume of $\frac{4}{3}\pi r_{\text{cut}}^3 \approx 4.189\ r_{\text{cut}}^3$ are fall in the cutoff distance and need to evaluate the interatomic distances. However, if a small cell edge is used, volume used to contain atoms whose interatomic distance need to be calculated will dramatically reduced. For example if $\frac{1}{2}r_{\text{cut}}$ is chosen as the edge of cells, the volume is $(2.5)^3\ r_{\text{cut}}^3$, only 57.87% of one in conventional method.

Furthermore, the edges of cells can be chosen to as small as at most one atom can be contained only. This treatment seems completely solve the problem of over-counting too many atoms, but actually it is not convenient to enumerate all neighbors for each cells, thus the overall performance is limited.

Compare with conventional Verlet table method, we can know that the advantage of cell-linked list method is the fast and efficient building of "neighbor list", in this method it is just assigning each atom to appropriate cells, and the disadvantage is that there are too many unnecessary atoms need to evaluate the interatomic distances in the "neighbor list" and the improved methods seem increase the complexity of constructing and using algorithm but the overall performance has little improvement only.

C. Improved method of Verlet table

In the section II-A one can known that the reason why the steps of maintaining Verlet table in conventional method is the order of $O(N^2)$ is enumerating every atoms in the simulation domain for finding out the neighbors of an atom. And also in section II-B, we know that cell-linked list method doesn't have this trouble so that its neighbor list constructing speed is higher, but the efficiency of utilizing neighbor list in force / potential evaluation is sacrificed. If the advantages of both methods can be combined together, the algorithm can be optimized.

In this work conventional Verlet table method and cell-linked list method are combined together, to prevent the constructing of the neighbor list table from over-counting too many atoms. Like the cell-linked list method, the whole simulation domain is partitioned into several cells, and the edges of these cells can be larger or smaller than the potential function cutoff distance r_{cut}, every time before constructing neighbor list table, each atoms are assigned to these cells by their coordinates, and then Verlet table search algorithm is used to construct the neighbor list table, but only atoms in neighbor cells are need to evaluate the interatomic distances, instead of all atoms in the system.

By considering the pipeline architecture of modern CPUs nowadays, further effort has been made to boost the computation performance: sorting the storage sequences of atoms in the memory and make atoms which in the same cell or neighbor cells are also in adjacent memory locations, thus the data can be loaded and cached more efficiently.

The overall procedure for constructing neighbor list table is shown in Algorithm 1.

Algorithm 1 Improved neighbor list algorithm

```
{Assigning all atoms into their appropriate cells}
for all atoms in the system do
  calculate the index $i$ of its appropriate cell;
  append the index of atom into the list of cell $i$;
end for
{Sorting atoms by their coordinates}
{Now carry out conventional Verlet table procedure}
for all atoms $i$ in the system do
  $l \Leftarrow$ the cell number of atom $i$
  for all cells $m$ among neighbors of cell $l$ and cell $l$ do
    for all atoms $j$ in cell $m$ do
      calculate interatomic distance $r_{ij}$;
      apply the periodic boundary condition;
      if $r_{ij} < r_{\text{cut}}$ then
        append $j$ into the neighbor list of atom $i$;
      end if
    end for
  end for
end for
```

III. Results

A molecular dynamics simulation program using Lennard-Jones $(12-6)$ two-body potential is developed to compare the performance of three different neighbor list algorithms. The benchmarks are carried out on Compaq Alpha Server DS20 with two EV67/667 MHz processors and Tru64 5.1A operating system installed, the program is written in Fortran 90 and compiled by Compaq Fortran compiler V5.5–1877. We also run the same benchmarks on a PC with one Intel Pentium III 866 MHz CPU and Red Hat Linux 8.0 installed, and a HP RX2600 workstation with two Intel Itanium2 900 MHz processors and Red Hat Linux Advanced Workstation installed. CPUs in these three platforms have different architectures: Alpha EV67 is a typical RISC (Reduced Instruction Set Computing) CPU, Intel Pentium III is a CISC (Complex Instruction Set Computing) CPU, and Itanium2 is declared as EPIC (Explicitly Parallelized Instruction Computing) architecture. The performances on three platforms differ largely, however the comparison between three different algorithms are qualitively similar. All data given in this section are results from Alpha Server DS20.

For measuring the performance quantitatively a new unit named *atom·step/second* is defined. It can simply calculated by multiply number of atoms and number of steps simulated divided by number of seconds elapsed. The larger is this value, the better is the overall performance.

In the simulation, some uniformly distributed Argon atoms with random locations are placed in the domain firstly, and the density of gas are predetermined. Then simulation in canonical ensemble is performed, and the number of steps is 10^2 for 10^4 atoms and above, or 10^3 for $10^3 \sim 10^4$ atoms, or 10^4 for 999 atoms or less. The Nośe–Hoover thermostat is used to implement the canonical ensemble simulation, and the temperature of system is 300 K.

To verify the improved Verlet table algorithm, all neighbor lists were dumped to the disk files and compared those in Verlet table algorithm for different system size. In the verification simulation the statistical quantities, such as total potential energy, total kinetic energy, transient temperature of system and the trajectory of atoms have been recorded for every 10 steps, and three sets of data generated from three algorithms are compared and ensure they differ in round-off errors only. A series of tests showed that neighbor lists from improved Verlet table method are exactly as same as those from conventional methods, and simulations with three different algorithms output exactly same results.

For different simulation the volume of system is increased with constant density, thus the number of atoms is increased correspondingly, and the performance is calculated.

Comparison of performances of molecular dynamics simulation with different algorithms are shown in Fig. 1 (single processor results) and Fig. 2 (dual-processor results)

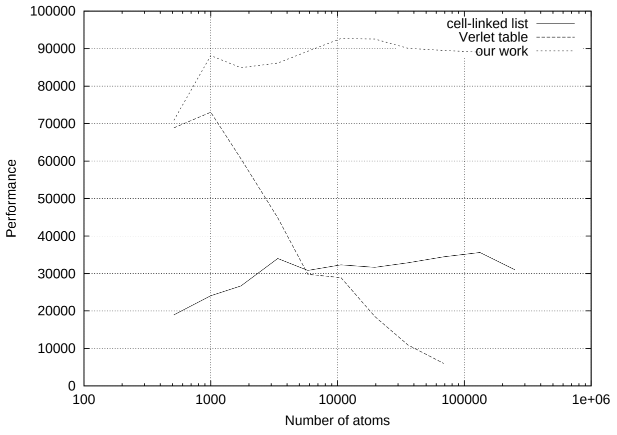

Fig. 1. Comparison of three algorithms on single processor system. The performance is measured in the unit of "atom·step/second", and its value can be calculated by multiply number of atoms by number of steps and divided by the whole simulation time. The three curves from top to bottom stand for performances of our improved method, Verlet table and cell-linked list method, respectively.

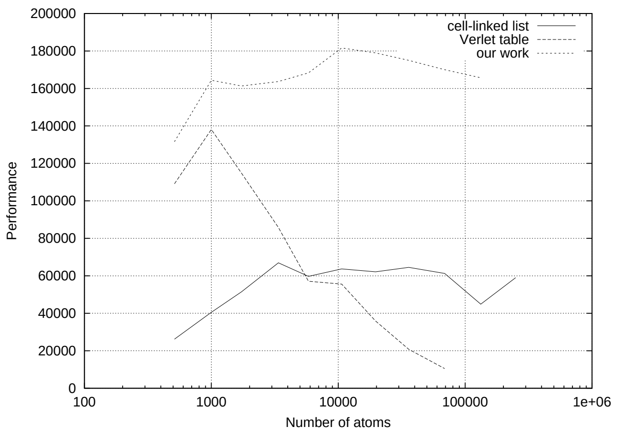

Fig. 2. Comparison of three algorithms on dual-processor system. The three curves from top to bottom stand for performances of our improved method, Verlet table and cell-linked list method, respectively.

IV. Conclusions

Nowadays the performance of CPU is increased follows Moore's law, and more and more powerful supercomputers are emerged continuously, thus larger and complicated molecular simulations will be attempted which involve larger amount of atoms and more complex potential functions. The expectation of running molecular simulation faster and easier for larger systems on existing platforms make it important to improve the conventional neighbor list updating algorithm in order to reduce the unnecessary interatomic distance calculations. A significant improvement of molecular dynamics simulation performance has been shown in this paper by improved order $O(N)$ Verlet table algorithm both on single processor platforms and dual-processor platforms. All results have shown that the new algorithm is superior than conventional Verlet table and cell-linked list algorithm in serial programs as well as parallelized programs.

Acknowledgments

This work was supported under the project "Heat transport in nanostructure — molecular dynamics and mode-coupling theory" of the Singapore-MIT Alliance, National University of Singapore.

References

[1] J. M. Thijssen, Computational Physics, Cambridge University Press, Cambridge, 1999.
[2] L. Verlet, Physics Review, **159**: 98–103 1967.
[3] D. Fincham, B. J. Ralston, Computer Physics Communication, **23**: 127–134, 1981.
[4] B. Quentrec, C. Brot, Journal of Computational Physics, **13**, 430–432 (1975); R. W. Hockney, J. W. Eastwood, Computer simulation using particles, McGrow–Hill, New York, 1981.
[5] M. P. Allen, D. J. Tildesley, Computer simulation of liquids, Oxford University Press, New York, 1990.
[6] W. Mattson, B. M. Rice, Computer Physics Communications, **119**: 135–148, 1999.
[7] J. H. Walther, P. Koumoutasakos, Journal of Heat Transfer, **123**: 741–748, 2001.