A Scalable Neighbor List Construction Method for Molecular Dynamics Simulations on Multi-core Platforms

Yali Liu[1, a], Wenyan Chai[1, b], Xiurong Li[2, c]

[1] School of Science and Technology, Beijing City University, Beijing, 100083, China

[2] College of Computer Science and Technology, Beijing University of Technology, Beijing, 100083, China

[a]email: liuyali@bcu.edu.cn, [b]email: cwycwyhd1@bcu.edu.cn, [c]email: xiurong_li@bjut.edu.cn

Keywords: Parallel Computing; Molecular Dynamics; Spatial Decomposition

Abstract. We present a scalable method to implement the construction of neighbor list in molecular dynamic simulations on multi-core platforms. Our methodology is based on the OpenMP programming model. Our method doesn't divide the whole of simulation region into cells. Each thread builds and maintains a few sliding cells for a neighbor list constructions. It reduces memory requirements and code complexity. Compared with some other methods applied in the neighbor list construction, the results of our experiments indicate that it is a scalable and efficient method.

Introduction

Molecular Dynamics (MD) simulation is an important methodology for detailed microscopic modeling on the molecular scale. Using Newton's equations of motion, the forces exerting on each particle in the MD simulations system can be computed as pair-wise interactions from other particles. There are some different approaches to computing interactions.

All-pairs method is the simplest to implement. It also is an inefficient method because all pairs of atoms must be examined whether or not the distance between them is less than a cut-off range r_c. Cell subdivision is commonly employed for organizing the information about atom positions [1]. The whole of simulation region is divided into small cells. All atoms are assigned to cells according to their current positions. The interactions are calculated between atoms that are either in the same cell or in immediately adjacent cells. Increasing code complexity and additional storage needed for a linked list are disadvantages of the cell method.

In most MD programs neighboring atoms that lie within a cut-off range r_c are stored in Verlet neighbor list [2]. In order to use this list over successive time-steps, r_c is replaced with r_n ($r_n = r_c + \Delta t$). Disadvantages are the additional storage needed for the neighbor list of pairs and the cost of construction the neighbor list. Therefore the research on high efficient parallel method used to build a neighbor list becomes increasingly important.

Research has been done for molecule dynamic simulations with shared-memory model on a multi-core platform [3-4]. In the creation of Verlet neighbor list, the cell method is common used [5]. But little study has been done for designing an efficient parallel implementation of a neighbor list constructions based on a shared-memory model in general multi-core processors [6-7].

In this paper, we proposed a scalable Thread-Private Sliding Cells (TPSC) approach for building a neighbor list in MD simulations. This method is based on the OpenMP programming model. In TPSC approach, spatial decomposition method is only used to assign atoms in a neighborhood into cells. In process of building the neighbor list, a few cells slide along a dimension with maximum range, therefore atoms assigned to cells will change, some atoms left cells and some atoms comes in. Compared with some other methods, the TPSC method achieves high performance.

TPSC method

Our TPSC method is based on spatial decomposition for building a neighbor list. It used the OpenMP programming model. TPSC method consists of the following steps:

1. TPSC method firstly finds a dimension with maximum range in a d-dimensional (d = 2 or 3) simulations system. The dimension is called MAXDIR. It also finds another dimension with bigger range in the d-dimensional simulations system. And the dimension is called SECONDDIR.

2. Then all atoms were sorted along MAXDIR and SECONDDIR dimensions. Results of sort were stored in array SortAtoms[]. It is obvious that the neighborhoods of atom i only involve those atoms whose coordinates are equal to or less than the coordinate of atom i. An atom that leaves the simulation region will be used of periodic boundary conditions. The sort process can be done in parallel. It isn't discussed in this paper.

3. After steps 1 and 2 have been done, all atoms can be partitioned and be evenly distributed among executing threads. N is the number of atoms, and P is the number of executing threads. A thread p (0 <= p < P) will build the neighbor list of those atoms whose array indices (in array SortAtoms[]) start at (N/P)×p and end by (N/P)×(p+1)-1. This partition method achieves better workload balance than simple one-dimmensional spatial decomposition.

4. The Last step is the central idea of TPSC method. For a thread p, work will be done on the following steps.

4.1) To first atom SortAtoms[(N/P)×p], search such atom whose distance in MAXDIR dimension with the first atom is just more than a distance r_n ($r_n = r_c + \Delta t$). The search starts from the current array index of the first atom along the negative direction of MAXDIR dimension. The founded atom index in array SortAtoms[] is assigned to a variable NS.

4.2) A small simulation region is constructed with those atoms whose array indices in array SortAtoms[] start at NS and end by (N/P)×p-1. The region is divided into small cells along SECONDDIR dimension, and that the cell highs all exceed $r_n/2$ in length. These atoms are assigned to cells according to their current positions. From Figure 1 we can see, to atom SortAtoms[i], it is obvious that its interactions are only possible between atoms that are in 5 neighboring cells according to the coordinate of the atom SortAtoms[i] in SECONDDIR dimension.

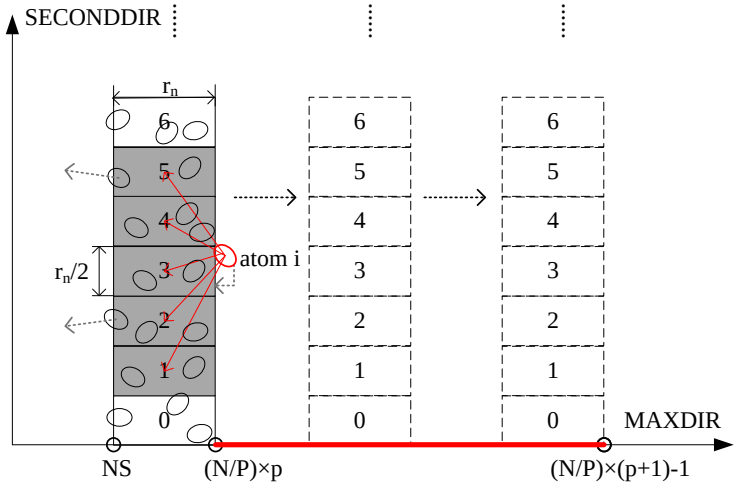


Fig.1. Sliding cells construct a small simulation region.

4.3) For each atom i ((N/P)×p <= i <= (N/P)×(p+1)-1), all atoms in 5 neighboring cells must be examined to generate a neighbor list of atom i, at the same time, atoms in 5 neighboring cells will be expelled from cells if whose distance in MAXDIR dimension with atom i are more than r_n. After the neighbor list of the atom i has been generated, atom i will be assign to one of 5 cells according to the coordinate of the atom i in SECONDDIR dimension. The cell is called sliding cell. Sliding cell, in the special form of a distance in MAXDIR dimension, contains atoms in the neighborhood of a given atom. We can see that a sliding cell is achieved by means of cell boundaries change in MAXDIR dimension. In process of building the neighbor list, cells slide along MAXDIR dimension separately.

If the cell subdivision method is used to generate Verlet neighbor list, the whole of simulation region will be divided into cells before the neighbor list is built, all atoms will be partitioned and assigned into cells using cell-linked lists. It led to workload and memory overload. Compared with

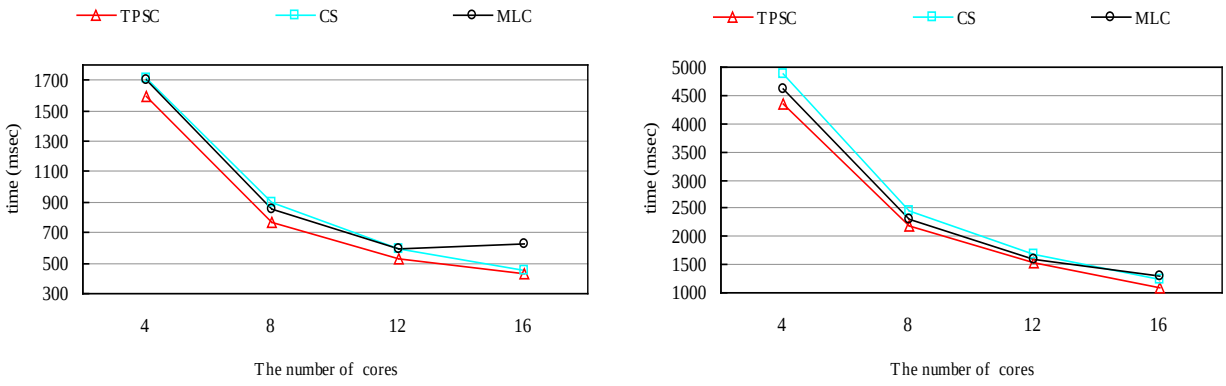
the cell method, TPSC method reduces the number of cells, avoids additional storage needed for the cell-linked list. Figure 2 shows the mainly parallel code of our TPSC method.

```
FindMaxdirSeconddir( );
Sort(N, MAXDIR, SECONDDIR, SortAtoms);  //parallel sort
double halfr = r_n / 2;
#pragma omp parallel private (i)
{
        int p = omp_get_thread_num();
        int start = (N/P)*p, end = (N/P)*(p+1);
        int NS = FindNeighStart(SortAtoms[(N/P)*p]);
        DivideCells(SortAtoms, head[], body[], SECONDDIR, NS, start-1);
        for (i = start; i < end; i++){
                int ipart = SortAtoms[i];
                int ccell = ( r[ipart][SECONDDIR] - MinCoord[SECONDDIR]) / halfr;
                for (int a = ccell - 2; a <= ccell + 2; a++){
                        Make_Periodic_Mirror(a);
                        for (int j = head[a]; j > 0; j = body[j]){
                                Make_Periodic_Mirror( );
                                if(|r[ipart][MAXDIR] - r[j][MAXDIR]| > r_n)
                                        ExpelAtomFromCells(j);
                                else if (|r[ipart] - r[j]| <= r_n)
                                        AppendNeigh(ipart,  j);
                        }
                        InsertAtomToCells(ipart, ccell);
                        AppendNeighList(ipart);
                }
        }
}
```

Fig.2. The mainly parallel code of TPSC method.

Experiments and Results

We performed experiments on a machine with 4 Intel Xeon(R) Quad-core E7320 processors and 16 GB memory. The operating system was Fedora release 9 with kernel 2.6.25. The compiler was gcc 4.3.0. Our experimental programs of MD application with EAM potential are written in C. The serial programs come from XMD code (which is a free MD program written by Jon Rifkin at the University of Connecticut). We modified the serial programs of building the neighbor list using our TPSC method described in Section 2. We also programmed its parallel programs using the Cell Subdivision (CS) method and the Modified Cell-Linked list (MCL) method (N_{div}=4). All of our execution times are based on an average for 20 separate times of the construction of the neighbor list, and the times include the construction of cell-linked list.



(a) On a medium-scale case. (b) On a large-scale case.

Fig.3. The times of the TPSC, the CS and the MLC methods.

We used two test cases in our experiments. The two cases were a medium-scale (1,062,882

particles) case and a large-scale (3,456,000 particles) case. In Figure 3, we gave the time results of the TPSC, the CS and the MCL methods, and that the CS and the MCL methods are based on two-dimensional cell.

Discussion

Firstly we compare the performances of the TPSC, the CS and the MLC methods on test cases. It is obvious that the TPSC method achieves highest efficiency, because the CS and the MLC method have more overhead of additional storage for cell-linked lists, which competes with cache space during the computations. And the TPSC method has good temporal and spatial locality because atoms in sliding cells are repeatedly accessed in memory. The MLC method gets better performance than the CS method when the number of executing cores is less than 16 on the medium-scale case, and it also gets better performance on the large-scale case.

Secondly we observe the scalability of our TPSC method. Figure 3 show that the performance of TPSC method has been improved with the increase in the number of cores and the increase in the number of atoms. Therefore our TPSC method is a scalable method for parallel building a neighbor list on multi-core machines. And it is expected that our method will continue to be scalable on future many-core architectures. Figure 3 (a) indicate that the MLC method is not a scalable method.

Conclusions

The research on high efficient parallel method used to build a neighbor list becomes increasingly important. We proposed a scalable Thread-Private Sliding Cells (TPSC) approach for building the neighbor list in MD simulations. In process of building the neighbor list, cells slide along a dimension with maximum range. Compared with some other methods applied in construction of a neighbor list, the results of our experiments indicate that TPSC approach is a scalable and efficient method.

Acknowledgement

Our work reported in this paper is supported by Youth Fund Project of Natural Science Foundation of Beijing (Grant No. 4144074).

References

[1] R. W. Hockney, S. P. Goel and J. W. Eastwood. Quiet High-Resolution Computer Models of a Plasma [J]. Journal of Computational Physics, 1974,14(2), 148-158.

[2] L. Verlet. Computer Experiments on Classical Fluids i. Thermodynamical Properties of Lennard-Jones Molecular [J]. Phys. Review, 1967,159, 98-103.

[3] Sadaf R. Alam, Pratul K. Agarwal, Scott S. Hampton, Hong Ong, and Jeffrey S. Vetter. Impact of Multicores on Large-scale Molecular Dynamics Simulations [C]. IPDPS. IEEE Press, 2008, 1-7.

[4] David J. Hardy, John E. Stone, and Klaus Schulten. Multilevel Summation of Electrostatic Potentials Using Graphics Processing Units [J]. Parallel Computing, 2009, 35(3), 164-177.

[5] DiBao Wang, FeiBin Hsiao, ChengHsin Chuang, YungChun Lee, Algorithm Optimization in Molecular Dynamics Simulation [J]. Computer Physics Communications, 2007, 177 , 551–559.

[6] William Mattson, Betsy M. Rice. Near-Neighbor Calculations Using a Modified Cell-Linked List Method [J]. Computer Physics Communications, 1999, 119(2), 135-148.

[7] Ulrich Welling, Guido Germano. Efficiency of Linked Cell Algorithms [J]. Computer Physics Communications, 2011, 182(3), 611-615.