PARALLEL MOLECULAR DYNAMICS SIMULATIONS OF DNA SYSTEMS

Alexander Lyubartsev and Aatto Laaksonen
Department of Physical Chemistry, Arrhenius Laboratory, University of Stockholm, S-10691 Stockholm, Sweden

I. INTRODUCTION.

Computer simulation methods, such as, Molecular Dynamics (MD) and Monte Carlo (MC) have now become important techniques to study fluids and solids. These methods provide a link between theory and experiment and they are also the only way to study complex many-body systems when both experimental techniques and analytical theories are unavailable.

The most time-consuming part (sometimes up to 99% of cpu time) of the MD simulations is the calculations of the forces $F_i(r_1, ...r_N)$. In a typical case of pair interactions the cpu time may scale with number of particles N between $O(N^2)$ and $O(N)$ depending on the algorithm and the type of interaction potential (note, that scaling as $O(N)$ may be reached only at a very large N and for systems with short-range interactions).

Now it is a standard routine procedure to simulate molecular systems consisting of 100-1000 particles, which in some cases (e.g. simple liquids) is sufficient to obtain a good description of corresponding macrosystem. For other systems, a larger number of particles is needed in order to describe them in a realistic way. Complex bio- and organic molecules (e.g. proteins, nucleic acids, membranes, carbohydrates, etc) immersed into a solvent increase the number of atoms to be involved into a simulation one to two or more orders. Also, larger the molecular systems grow, longer simulations are needed to follow low-amplitude motions and slow conformational transitions. It is clear that the rate of the progress towards more complex molecular models is set, to a large extent, by advances in microprocessor technology and computer architecture.

Computer simulations of many-particle systems are well suited for parallel computer systems. The basic reason for this is that the forces acting on each particle can be calculated independently in different processors. However, the most optimal parallel scheme for a particular problem depends both on the hardware in hand and on the system under investigation (size, type of interaction, etc.). Electrostatic interactions, fast intramolecular motions due to explicit modeling of hydrogens, angle and torsional angle forces of macromolecules - all these kinds of forces require a special treatment to create an effective parallel code. The present project has the aim to develope a software to simulate systems of complex molecules (including DNA solutions) with number of atoms $10^3 - 10^4$ during a possibly longer time (up to several nanoseconds) using several or several dozens connected processors.

II. THE MOLECULAR DYNAMICS ALGORITHM.

A. General organization.

Complex molecular systems are often described by force fields like AMBER [1] or Gromos [2].These force fields contain terms for following interactions:

1) atom-atom short-range interactions (Lennard-Jones potential)
2) atom-atom electrostatic interactions
3) intramolecular interactions: covalent bonds, covalent angles and torsional angles.

In principle, calculations of atom-atom interactions require a double sum over all the atom pairs. The application of a cut-off radius for the non-bonded interactions allows to considerably reduce the cpu time for calculation of atom-atom interactions, by effectively setting the interactions between particles separated by distances larger than the cut-off to zero.

Two problems, however, emerge here. First, the electrostatic interactions are long-ranged, and strictly, no cut-off without a special treatment can be applied. Second, in order to decide wether to calculate the forces between a given atom pair or not, one still should know the distances between all atomic pairs, or at least to have a list of atom pairs with distances less than the cut-off (list of neigbours).

One of the most effective and popular methods of treatment of electrostatic interactions is the Ewald summation method [3]. The Ewald method splits up the total force into a long-range and short range component. The long-range part is calculated in the reciprocal space while the short-range part is treated alongside with the Lennard-Jones forces. The convergence of the two parts of the Ewald sum is regulated by a parameter. The optimal choice of the convergence parameter leads to scaling of the cpu time as $O(N^{3/2})$.

Creation of the neigbour lists is another problem. In liquids this list should be updated periodically. In the linked-cell method the search of neighboring pairs can be limited only by the current and the touching cells; this leads to an $O(N)$ algorithm. However the true $O(N)$ algorithm is achieved only at very large N; for the "average size" systems of $10^3 - 10^4$ particles periodical (e.g. each 10 MD steps) update of neigbours list by looking through all the atom pairs occurs effective enough. Though cpu time of this block is scaled as $O(N^2)$, the coefficient is small, and for example in a test run with 2000 H_2O molecules (6000 atoms) this part of the program consumes about 6% of the cpu time.

Calculations of intramolecular forces due to covalent bonds, angles and torsional angles are straightforward, allthough the the resulting expressions may be complicated.

Irrespectively to the parallelization, an essential saving of cpu time may be achieved by applying the multiple time scale algorithm. Different kinds of forces in the system fluctuate at different, characteristic time scales. In systems with explicit treatment of hydrogens, covalent bond forces, Lennard-Jones and electrostatic interactions at short distances between atoms require an updated force calculation after 0.2-0.4 fs, whereas long range parts of Lennard-Jones and electrostatic interactions, which consume most of cpu time, may be calculated after 2-4 fs. In the present program the two time scale algorithm is applied [4]. The covalent forces and atom-atom forces for atoms closer than 5Å are calculated at every short time step. Forces between atoms with distance from 5Å to cut-off and reciprocal part of the Ewald sum are calculated at every long time step. Correspondingly, two lists of neigbours are calculated in the program - for fast and for slow forces.

In the present algorithm a SIMD scheme (single instruction - multiple data) is applied. All nodes have the same program code, but it is executed on different data. The "core" of the MD algorithm: preparation of the initial state, moving of particles, average collection, output - that takes only a minor cpu time - is executed on one (zero-th) node. After each MD step new coordinates and velocites of particles are transfered to the other nodes for force calculations.

B. Parallelization of specific parts of the program

For an effective parallellization the whole list of atom pairs $(I, J = 1, ...N, I < J)$ should be uniformly divided between the available number of nodes. The condition $I < J$ is set to avoid a double count, but it makes difficult to divide equally atom pairs between the nodes. To avoid this, the following scheme was applied. All particles are divided equally between procesors (loop "do I=TASKID,N,NUMTASK" , where TASKID - node number, NUMTASK - total number of available nodes). For each I we have an inner loop over J with the following condition: $I - J$ is even for $I > J$ and $J - I$ is odd for $I < J$. So the atom pairs occur uniformly distributed between nodes and each one is treated only once. Then for each node lists of closest ($r <$ 5Å) and far ($5 < r < R_{cut}$) neigbours are calculated. These lists are local for each node and are recalculated after about 10 long time steps. They are used for LJ and electrostatic (real space part) force calculations. Using of local lists saves memory and make it possible to use the same code both for scalar and for parallel execution.

Parallelization of the reciprocal part of the Ewald sum is more or less straigtforward. This contribution is expressed as a sum over reciprocal space vectors. Each node calculates force contributions due to a certain (fixed for this node) group of reciprocal vectors.

Calculations of forces due to covalent bonds, angles and torsions can be done independently for each bond, angle, or dihedral angle. At the begining of simulations all the bonds, covalent angles and torsions are distributed uniformly over available nodes. Each node gets its own list of bonds, angles and torsions and corresponding contributions to the forces and energies are calculated in parallel.

Different forces acting on each atom from other atoms are accumulated first on each node separately. After completing the calculations on each node all the forces acting on a given atom are transfered to the node cresponding to this atom and summed up. Then the atoms are moved according to a chosen integration scheme (in the present project three versions of MD are implemented: constant energy (NVE), constant temperature (NVT) and constant pressure and temperature (NPT) molecular dynamics). After completing an MD step new positions and velocities of atoms are broadcasted to all nodes.

C. Program timing.

We have tested the program for simulation of a DNA fragment in water solution. The system consists of a DNA periodical fragment containing 635 atoms, 1050 water molecules and 20 Na+ ions. Table 1 shows cpu time for all main blocks of the program (per 100 long MD steps).

TABLE I. CPU time (in sec) of different parts of the program for simulation of the DNA-water-ions system (per 100 MD steps and per node)

Num.of nodes	Intermolec. forces	Ewald	Intramolec. forces	update list of neighbours	moves	data transfer	total	speedup
1	1006	262	29	92	14	0	1417	1
2	504	135	14.8	46	7	16	713	1.97
3	347	93	10.3	31	5	18	517	2.7
4	255	68	8.4	23	4	19	390	3.5
5	209	54	6.7	18	3	20	332	3.8
8	145	33	4.6	12	2	20	227	5.3
10	110	27	3.2	9.5	1.8	22	178	5.9
13	81	21	2.8	7.2	1.6	24	129	7.5
16	66	15	2.4	4.8	1.4	25	107	8.5
32	33	8	1.2	2.5	0.8	28	72	10.

In the presented algorithm the cpu time for calculations of all the forces and for the update of neighbours lists scaled as $1/M$ with the number of processors M. Time for particle moving doesn't change because this part is not parallelized, but even on 32 processors this part consumes less than 10is proportional to $ln(M)$. As a results of this the speedup is not linear as the number of processors is increased.The increasing overhead due to the growing data transfer slows down the speed of the calculations as the number of processors is increased.

REFERENCES:

1. S.J. Weiner, P.A. Kollman, D.T. Ngyuen, and D.A. Case, J. Comput. Chem., 7, 230 (1986)
2. W.F. van Gunsteren and H.J.C. Berendsen, Gromos Library Manual, Biomos Nijenborgh 16, Groningen, The Netherlands
3. P. Ewald, Ann. Phys., 64, 253 (1921)
4. M. Tukerman, B.J.Berne and G.J.Martyna, J.Chem.Phys., 97, 1990 (1992)