Parallel Molecular Dynamics Simulations of Biomolecular Systems

Alexander Lyubartsev and Aatto Laaksonen

Department of Physical Chemistry, Arrhenius Laboratory,
University of Stockholm, S-10691 Stockholm, Sweden

Abstract. We describe a general purpose parallel molecular dynamics code, for simulations of arbitrary mixtures of flexible molecules in solution. The program allows us to simulate molecular systems described by standard force fields like AMBER, GROMOS or CHARMM, containing terms for short-range interactions of the Lennard-Jones type, electrostatic interactions, covalent bonds, covalent angles and torsional angles and a few other optional terms. The state-of-the-art molecular dynamics techniques are implemented: constant-temperature and constant-pressure simulations, optimized Ewald method for treatment of electrostatic forces, double time step algorithm for separate integration of fast and slow motions. The program is written in standard Fortran 77 and uses MPI library for communications between nodes. The scalable properties of the program do not depend on the complexity of the studied system and are determined mainly by the hardware and communication speed. Examples of a few molecular systems differing by the composition will be given: Ionic water solutions, large DNA fragments in water solution with counter ions, a phospholipid membrane system.
Keywords: parallel algorithms, molecular dynamics, computer simulations

1 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 MD method provides a numerical solution of classical (Newton's) equations of motion:

$$m_i(d^2 r_i/dt^2) = F_i(r_1, ...r_N) \tag{1}$$

where force $F_i(r_1, ...r_N)$ acting on particle i is defined by the interaction potential (or force field) $U(r_1, ..., r_N)$:

$$F_i(r_1, ...r_N) = -\frac{\partial}{\partial r_i} U(r_1, ..., r_N) \tag{2}$$

The most time-consuming part (sometimes up to 99% of the 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 order of 100-1000 particles, which in some cases (e.g. simple liquids) is sufficient to give 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 involved atoms one to two or more orders of magnitude. 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 as well as by development of appropriate software.

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.

We have developed a general purpose molecular dynamics code (MDynaMix) [1] for simulations of arbitrary mixtures of rigid or flexible molecules employing the most modern simulation techniques: double time step algorithm for fast and slow modes, optimized Ewald method for electrostatic interactions, constant temperature - constant pressure algorithm. The program can be used for simulation of mixtures of molecules interacting with AMBER- or CHARMM-like force fields [2], [3].The code is highly universal and well suitable for simulation of both simple molecules and complex biological macromolecules. The code uses only standard Fortran77 statements, being able to run on any parallel system with MPI-library installed. In the latest version (4.2) additional features were included: separate pressure control in different directions (for simulation of anisotropic systems), generalized reaction field method for electrostatic interactions, truncated octahedron or hexagonal simulation cell, parallel SHAKE algorithm for constrained dynamics, different types of torsional angle potentials and a few other options. Below we describe details of the program organization, parallelization algorithm and performance.

2 Parallel Molecular Dynamics

2.1 General Organization

Complex molecular systems are often described by force fields like AMBER [2] or CHARMM [3]. 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.

4) optional terms (hydrogen bonds, anharmonic bond potential, other specific interactions)

The general functional form of such a force field is:

$$
\begin{aligned}
U(r_1, ..., r_N) = \sum_{i<j=1}^{N} \left(\frac{A_{ij}}{r_{ij}^6} + \frac{B_{ij}}{r_{ij}^{12}} \right) + \sum_{i<j=1}^{N} \left(\frac{q_i q_j}{r_{ij}} \right) + \\
\sum_{bonds} K_{bond}(r_{ij} - r_{ij}^0)^2 + \sum_{angles} K_{ang}(\varphi_a - \varphi_a^0)^2 + \\
\sum_{torsions} K_{tors}(1 + Cos(m_t \psi_t - \psi_t^0)) + U_{optional}
\end{aligned} \tag{3}
$$

where r_{ij} is the distance between atoms i and j, and other constants define force field parameters for each chemical atom type.

In principle, calculations of pairwise atom-atom interactions (Lennard-Jones and electrostatic forces, i.e the first and the second term in expression (3)) require a double sum over all the atom pairs. Usually this is the most time consuming part of the force calculations. The application of a cut-off radius for these interactions allows one to considerably reduce the cpu time, by effectively setting the interactions between particles separated by distances larger than the cut-off distance 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. Attempts to simulate molecular systems with electrostatic interactions using simple spherical cutoff, even of a rather large radius, may lead to serious artifacts [4]. Second, in order to decide whether to calculate the forces between a given atom pair or not (e.g whether the atoms are within or out the cutoff radius), 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 neighbours).

One of the most effective and popular methods of treatment of electrostatic interactions is the Ewald summation method [5]. 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 by the Fourier transform, 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 a scaling of the cpu time as $O(N^{3/2})$. The optimized Ewald method is implemented in the program. Recently a new version of the Ewald method (Particle Mesh Ewald [6]) has appeared, which scales as $NlnN$ for large $N > 10^4$. We are planning to implement the Particle Mesh Ewald method in a future program release.

Creation of the neighbour 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. Although the cpu time of this block scales 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 5% of the cpu time.

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 [8]. 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 neighbours are calculated in the program - for fast and for slow forces.

2.2 Parallelization Strategy

There are two main strategies in parallelization of molecular dynamics programs, which called "Replicated data" and "Domain decomposition". In the replicated data method all the nodes know positions of all the particles in the system, while calculation on different contributions to forces goes in parallel. The advantages of this method is a relative simplicity to distribute force calculations over processors: different contributions to the sum (3) have a form of a sum of a large number of similar independent terms (i.e. sums over atom pairs, covalent bond, covalent angles, etc). Such calculations can be done in parallel on different processors equally effective for virtually all kinds of the force fields and molecular structures. The weak point of the replicated data approach is that it requires relatively extensive communications between processors, to collect data on contribution to the forces from all the nodes and to distribute new particle's positions to all the nodes. In the "Domain decomposition" approach particles are distributed between the nodes, usually each node being responsible for the particles in the corresponding subcell. The communication overhead may be lower in this case comparing to the replicated data method, but other sources of overheads arise due to necessity to monitor moving of particles through the subcells, to implement Ewald summation of electrostatic interactions, to get use of the Newton's 3-rd law, etc. An addition of extra functionality to the program in the domain decomposition method may also lead to essential increase of cpu time both for calculations and for communications.

It is believed that the domain decomposition method is more effective for massively parallel computer systems to run simulation programs for particles interacting with simple short range potentials; for parallel simulation of complex biomolecular systems on computers with several or several dozens processors the domain decomposition method is preferable. In our molecular dynamics program, the replicated data method is implemented.

2.3 Parallelization of Specific Parts of the Program

For an effective parallelization of non-bonded interactions (first and second sum in (1))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 processors (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 ($r < 5$Å) and far ($5 < r < R_{cut}$) neighbours are calculated. These lists are local for each node and are recalculated after about 10 long time steps. They are used for the Lennard-Jones and electrostatic (real space part) force calculations. Using of local list of neighbours on each node makes it possible to use the same code both for parallel and single-processor architecture. Moreover, since the list of neighbours is the biggest data structure in the program, distributing it over processors reduce the total memory requirements, or makes it possible to simulate much bigger molecular systems with the same memory limit per processor.

Parallelization of the reciprocal part of the Ewald sum is more or less straightforward. 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 beginning 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 corresponding to this atom and summed up. The whole operation is made by a single MPI call MPI_REDUCE_SCATTER, which, depending on the implementation, may work essentially faster than consecutive summing up forces (REDUCE) followed by their distribution over the nodes (SCATTER). Then on each node the atoms are moved according to a chosen integration scheme: constant energy (NVE), constant temperature (NVT) or constant pressure and temperature (NPT) molecular dynamics [7]. The double time step algorithm [8] is easily integrated into this scheme. After completing an MD step new positions of atoms are broadcasted to all nodes.

2.4 Portability

The program is written in standard Fortran 77 and highly portable, being able to run on any parallel system with MPI-library installed. It can be also easily made run on a single-processor computer. So far the program was tested on parallel systems IBM SP2, Cray T3E, PC-linux clusters employing MPI calls through an ethernet network, as well as on single processor systems IBM RISC 6000, DEC Alpha, PC-linux. Porting the code to another computer system normally requires only small changes of the Makefile.

3 Examples

The performance of the program was tested in details on several molecular systems:

1. A periodic fragment of DNA in ionic aqueous solution: one helical turn of DNA (635 atoms), 1050 water molecules, 30 Na and 10 Cl ions (3825 atoms totally). In other simulations Na ions was substituted by Li or Cs ions.

2. An NaCl ion solution: 20 NaCl ion pairs and 1960 water molecules, 5920 atoms totally.

3. A lipid bilayer. 64 DPPC lipid molecules (50 atoms each) and 1472 water molecules, 7616 atoms totally.

The three molecular systems are qualitatively different on the composition: the first one contains one macromolecule in solution, the second is a mixture of small molecules, the third one is an inhomogeneous system consisting of a number of middle size molecules and waters.

The dependence of the total computation time on the number of nodes in each case is given in **Table 1**. In all the cases the double time step algorithm with a short step of 0.2 fs and a long step of 2 fs was used. The long-range electrostatic interactions were taken into account by the Ewald method. The cut-off in the reciprocal space was determined from the conditions that the remaining terms were less then 0.001 relative to the total electrostatic energy.

The calculations were performed on IBM SP2 Strindberg at PDC, KTH, and on CRAY T3E in the National Supercomputer Center, Linköping. The program shows good scaling properties in all cases running on up to 32 processors.

Table 1. CPU time (in min) for 1 ps simulation of 3 different molecular systems depending on number of processors. Box sizes and cutoff distance R_{cut} are given in Å

				nodes						
System	Num.atoms	Box sizes	R_{cut}	1	2	4	8	16	32	64
					IBM	SP2	160	Mhz proc.		
1. DNA	3825	33x33x34	13	78	39	20	11	6	3.8	2.5
2. Ion solution	5920	39x39x39	13	138	69	35	18	9.2	5.	3.6
3. lipid bilayer	7616	44x44x64	14	187	94	48	25	13	7.5	5.5
					Cray		T3E			
1. DNA	3825	33x33x34	13	121	60	31	16	9	5.3	3.5
3. lipid bilayer	7616	44x44x64	14	268	134	70	37	19	11	7
					Pentium	-2	300	Mhz		
3. lipid bilayer	7616	44x44x64	14	250	126	-	-	-		

We have used the MDynaMix program for practical simulations of these molecular systems on the nanosecond time scale. Moreover, for the DNA solution simulations were carried out with several different counterion types. Both structural and dynamical properties of the molecules are studied. The reader is referred to the corresponding articles for details [9], [10].

Acknowledgements

We are grateful to the Center for parallel Computing at the Royal Institute of Technology (Stockholm) and National Supercomputing Center (Linköping) for granting the computer facilities, and the Swedish National Science Foundation (NFR) and the Foundation for Strategic Research (SSF) for financial support.

References

1. http://www.fos.su.se/physical/sasha/md_prog.html
2. Weiner, S.J., Kollman, P.A., Ngyuen, D.T., Case, D.A.: An All Atom Force Field for Simulations of Proteins and Nucleic Acids, J. Comput. Chem. **7** (1986) 230-252.
3. MacKerell, A.D., Wiorkiewicz-Kuczera, J., Karplus, M.: An All-Atom Empirical Energy Function for the Simulation of Nucleic Acids, J.Am.Chem.Soc. **117** (1995) 11946-11970.
4. Tironi, I.G., Sperb, R., Smith, P.E., van Gunsteren, W.F.: A Generalized Reaction Field Method for Molecular Dynamics Simulations. J. Chem. Phys., **102** (1995) 5451 - 5458.
5. Allen, M.P., Tildesley, D.J.: *Computer Simulation of Liquids*, Clarendon: Oxford, 1987
6. Darden, T., York, D., Pedersen, L.: Particle Mesh Ewald: A $NlnN$ Method for Ewald Sums in Large Systems, J. Chem. Phys., **98** (1993) 10089-10092.
7. Martyna, G.I., Tuckerman, M.E., Tobais, D.J., Klein, M.L.: Explicit Reversible Integrators for Extended Systems Dynamics, Mol. Phys., **87** (1996) 1117-1157.
8. Tuckerman, M.E., Berne, B.J., Martyna, G.J.: Reversible Multiple Time Scale Molecular Dynamics, J. Chem. Phys., **97** (1992) 1990-2001.

9. Lyubartsev, A.P., Laaksonen, A.: Concentration Effects in Aqueous NaCl Solutions. A Molecular Dynamics Simulations. J. Phys. Chem., **100** (1996) 16410-16418.
10. Lyubartsev, A.P., Laaksonen, A.: Molecular Dynamics Simulations of DNA in Presence of Different Counterions. J. Biomolec. Structure and Dynamics (in press).