

Molecular Dynamics Algorithms for Massively Parallel Computers

Norbert Attig¹, Marius Lewerenz²,
Godehard Sutmann¹, and Reiner Vogelsang³

¹ Central Institute for Applied Mathematics,
Forschungszentrum Jülich, 52425 Jülich, Germany
E-mail: {n.attig, g.sutmann}@fz-juelich.de

² LADIR/Spectrochimie Moléculaire, Université Pierre et Marie Curie,
4, Place Jussieu, 75252 Paris Cédex 05, France
E-mail: lewerenz@spmol.jussieu.fr

³ Silicon Graphics GmbH, Professional Services,
Am Hochacker 3, 85630 Grasbrunn, Germany
E-mail: reiner@munich.sgi.com

A new modular program system, DMMD (Distributed Memory Molecular Dynamics) has been designed which employs modern elements of the Fortran 90 language for abstraction and encapsulation. The program uses the message passing paradigm which is applicable to both distributed and shared memory systems. Communication is handled in a special interface library which efficiently translates high level Fortran 90 concepts into the more primitive addressing mechanisms of current message passing protocols. Furthermore it allows transparent switching between message passing libraries. This makes it a powerful, flexible tool for parallel Fortran 90 applications. The modular concept of DMMD allows easy modification and extensions of its capabilities towards new interaction models or integrators. The design has been tested in a serial version which also serves as a reference point for the parallel version which will be used to benchmark several strategies for data and work distribution. We report first results obtained with DMMD on a Cray T3E system for a simple Lennard-Jones mixture and comparisons with other massively parallel molecular dynamics codes.

1 Introduction

Within the John von Neumann Institute for Computing (NIC) the Central Institute for Applied Mathematics (ZAM) acts as a national supercomputer centre and provides access to high performance computer resources and programming expertise to a wide scientific community.

⋮

These applications will be outlined in Sec. 2. Strategies for the parallel evaluation of forces are discussed in Sec. 3. The design concept of the DMMD molecular dynamics program is described in Sec. 4 and the key features of a general Fortran 90 communication interface and other software engineering considerations are the subject of Sec. 5. A description of important features of the main hardware platform, CRAY T3E, and benchmark results are given in Secs. 6 and 7, respectively.

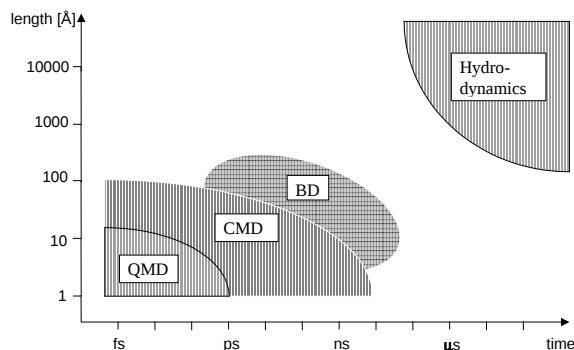


Figure 1. Schematic picture of the application domains of quantum molecular dynamics (QMD), classical molecular dynamics (CMD), Brownian dynamics (BD) and hydrodynamics on time and spatial scales.

2 Motivation

One challenge in modern high performance computing is the development of programs which are able to simulate systems on a microscopic basis, i.e. atomic resolution, on *large* time and spatial scales, where *large* means micrometers and microseconds. This will open the field to treat problems like crystal growth, crack propagation or protein folding which could, until now, only be described with various simplifications. Fig. 1 gives a schematic overview over different simulation methods and their range of applicability.

⋮

3 Parallel Force Evaluation in Molecular Dynamics

This section describes different types of algorithms for short range interactions in parallel molecular dynamics calculations. The two types of algorithms which we discuss in detail are the particle decomposition (PD) and spatial decomposition (SD) methods. The characteristics of these algorithms are in principle very different and they have their own advantages depending on the problem which is to be solved.

⋮

3.1 Neighbour Lists

Each MD step requires the calculation of the forces acting on a particle due to all other particles in the simulation box.

⋮

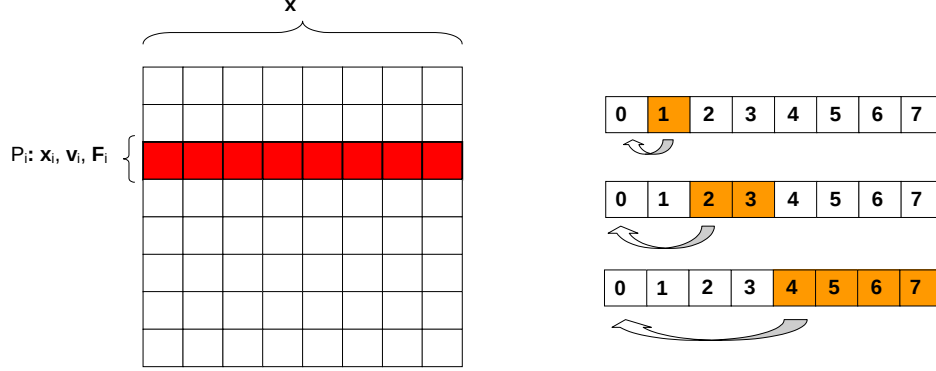


Figure 2. Example for the PD1 algorithm on 8 processors. Left: The force loop requires the whole vector $\mathbf{x}$ on all processors. The equations of motion are solved on one processor with a subset $\mathbf{x}_i, \mathbf{v}_i, \mathbf{F}_i$ on processor PE_i . Right: The procedure to copy the whole vector $\mathbf{x}$ on one PE requires only $\log_2(P)$ steps.

3.2 Particle Decomposition Method

PD1: In this algorithm each processor stores the data of the coordinates, velocities and forces of a subset of particles $N_p = N/P$, where N is the total number of particles of the system and P is the number of processors. It is assumed here and in the following that $\text{mod}(N/P) = 0$ but it is easy to generalise the algorithm to non-homogeneous distributions. In order to test for mutual interactions between particles the coordinates of all particles, stored on other processors, are communicated to the local PE. The most natural way to realise the communication would be to perform on each processor $P-1$ send/receive operations so that the coordinates of all particles in the system are duplicated on each PE. A more efficient way which needs only $\log_2(P)$ send/receive operations, was proposed by Fox et al.[1] and is illustrated in Fig. 2.

⋮

With this convention one may subdivide the force loop into loops of equal length through

```
offset = 0
do i j = 1, n_loop
  offset = offset + 1
  do i = 1, N_p
    j = i + offset
     $\delta\mathbf{x} = \mathbf{x}_i - \mathbf{x}_j$ 
    ⋮      (force evaluation)
  enddo
enddo
```

where $n_{loop} = (N_p - 1)/2$. The performance of PD1 is summarised in Tab. 1.

1. All-to-all communication of coordinates: local copy of total vector $\mathbf{x}$ on each PE	$N - N_p$
2. Construction of neighbour lists: if ($L < 4R_c$) Verlet list	$N_p(N - 1)$
if ($L \geq 4R_c$) linked cell list	N_p
3. Calculation of force vector for $i \in N_p$	N_p
4. Update particle positions	N_p

Table 1. Scaling behaviour with number of particles for PD1.

4 Molecular Dynamics Program Design

5 Software Engineering

6 Hardware Considerations

The DMMD program runs on workstations (SUN, IBM R50, DEC) as well as on powerful massively parallel processor (MPP) systems like CRAY T3E and Origin 2000. Program development is done on workstations, while full production runs with large numbers of particles are performed on MPP systems only; benchmarks and test simulations run on all platforms available. Target platforms for DMMD are primarily distributed memory MPP systems, e.g. the CRAY T3E, which have the following advantages over shared memory systems: (i) programs have the most general structure, i.e. algorithms and methods developed for distributed MPP systems are also applicable to shared memory machines, which is not true *vice versa*! (ii) the CRAY T3E is currently one of the most powerful supercomputers available^a, and it is especially suited to tackle large-scale MD problems. This was demonstrated by *world record simulations* of the groups of Trebin[2] and of Kollman[3].

⋮

7 Benchmarks

7.1 Proper MD Programs for Comparison

The simplest benchmark programs available for our purpose are the ones from Steve Plimpton which are freely available on the web[4]. They simulate Lennard-Jones mixtures with different parallelisation strategies: particle distribution (PD1, PD2), domain (SD) and force decomposition[5].

^aTwo large T3E systems are installed at the Forschungszentrum in Jülich. The first one is equipped with 512 processors of type T3E-600 and a main memory of 128 MByte per PE. The second one consists of 512 processors of type T3E-1200E with 512 MByte main memory per PE.

LAMMPS (developed at Sandia National Laboratory within the CRADA project[6]), a pure MD program, partially contains these algorithms. It is designed to model large molecular systems and it is also described on the web.

⋮

7.2 The Benchmark Run

Our benchmark problem is a simulation of a simple Lennard-Jones fluid with the parameters given in Tab. 2.

⋮

•	System of short range interacting particles:
	Argon-Krypton mixed Lennard-Jones (LJ) fluid
•	NVE ensemble
•	Global physical parameters:
density:	$n = 0.0181 \text{ \AA}^{-3}$
temperature:	$T = 116 \text{ K}$
LJ parameters:	$\sigma_{\text{Ar}} = 3.405 \text{ \AA}, \epsilon_{\text{Ar}} = 119.8 \text{ K}$
	$\sigma_{\text{Kr}} = 3.633 \text{ \AA}, \epsilon_{\text{Kr}} = 167.0 \text{ K}$
mass:	$m_{\text{Ar}} = 39.948 \text{ amu}, m_{\text{Kr}} = 83.8 \text{ amu}$
cutoff radius:	$r_{\text{cutoff}} = 2.5 \sigma$
skin radius:	$r_{\text{skin}} = 0.5 \sigma$
time step:	$\Delta t = 20 \text{ fs}$
•	Global simulation parameters:
system:	CRAY T3E-1200, streams enabled

Table 2. Benchmark Parameters

7.3 Further Benchmark Remarks

In a real production run neighbour lists are obligatory. The list update has only to be performed in certain time intervals, depending on the skin radius and the diffusivity of the system. DMMD does not yet contain neighbour lists but we have tested this technique with Plimptons' kernels where additionally Newton's third law is not applied (algorithm PD1) and neighbour lists are updated at regular intervals of 20 time steps. Tab. 3 reports the results of these runs.

First of all, the table clearly shows that domain decomposition is faster than particle distribution, nearly independent of the number of particles.

⋮

	N = 2048				N = 97556			
	particle decomposition		domain decomposition		particle decomposition		domain decomposition	
	Verlet	linked cell	Verlet	linked cell	Verlet	linked cell	Verlet	linked cell
#PEs	time/step/N [μ s]				time/step/N [μ s]			
1	49.31	41.71	20.65	17.63	927.9	39.10	761.2	16.94
2	24.81	21.01	12.74	10.30	463.3	19.54	245.4	8.87
4	12.67	10.76	7.34	6.01	231.9	9.88	78.8	4.68
8	6.57	5.61	4.19	3.56	116.1	5.09	25.5	2.44
16	3.55	3.07	2.21	2.10	56.7	2.66	8.0	1.28
32	2.09	1.86	1.25	1.29	29.3	1.51	2.6	0.66
64	1.34	1.20	0.82	0.82	14.8	0.86	0.8	0.34
128	1.00	0.94	0.62	0.64	7.5	0.55	0.3	0.19

Table 3. Plimpton-Benchmarks with 2048 and 97556 particles

8 Concluding Remarks

We have described a molecular dynamics package for systems with short range interactions. The program is written in a strong typing Fortran 90 dialect and designed in a strictly modular form. A message passing wrapper has been developed which supports different message passing libraries, e.g. MPI, and avoids common problems between Fortran 90 and communication libraries. Several types of algorithms for short range interactions were discussed and the performance of the program was compared to Plimpton's benchmark kernels[5].

⋮

Acknowledgements

We are indebted to G. Groten for many fruitful discussions about the intricacies of Fortran 90 and for the coding of a significant part of the interface library. The benchmark computations were performed with a grant of computer time provided by the VSR of the Forschungszentrum Jülich.

References

- [1]G. Fox, M. Johnson, G. Lyzenga, *et al.*, *Solving problems on concurrent processors vol. 1: General techniques and regular problems*, Computer in Physics, **3**, no. 1, pp. 83–84, Jan. 1989, ISSN: 0894-1866. DOI: 10.1063/1.4822815.
- [2]*Md simulation of an fcc lattice with 1.213.857.792 atoms*. [Online]. Available: <http://www.itap.physik.uni-stuttgart.de/~joerg/imd.html>.
- [3]L. W. Y. Duan and P. A. Kollman, *Md simulation of a 1 μ s trajectory of a small protein in water*, Proc. Natl. Acad. Sci. USA, **9**, p. 9897, 1998.

- [4][Online]. Available: <http://www.cs.sandia.gov/~sjplimp/main.html>.
- [5]S. Plimpton, *Fast parallel algorithms for short-range molecular dynamics*, Journal of Computational Physics, **117**, no. 1, pp. 1–19, 1995, ISSN: 0021-9991. DOI: <https://doi.org/10.1006/jcph.1995.1039>.
- [6][Online]. Available: <http://www.cs.sandia.gov/~sjplimp/crada.html>.