International Journal of Modern Physics C, Vol. 11, No. 2 (2000) 317–322

A MOLECULAR DYNAMICS RUN WITH 5 180 116 000 PARTICLES

J. ROTH, F. GÄHLER, and H.-R. TREBIN

Institut für Theoretische und Angewandte Physik, Universität Stuttgart
Pfaffenwaldring 57, 70550 Stuttgart, Germany
E-mail: *johannes@itap.physik.uni-stuttgart.de*

Received 17 December 1999
Revised 22 December 1999

We report on the development of IMD, a scalable program for classical molecular dynamics simulations on massively parallel supercomputers. New features like online-visualization and metacomputing are described.

Keywords: World Record; Metacomputing; Online Visualization.

1. Introduction

The need for large-scale molecular dynamics simulations is increasing in many fields. For example, in crack propagation studies there is the multiscale problem. Although the interactions may be short-ranged, the strain fields generated by the crack tip are not. Sound waves emitted from the crack tip may be reflected at the boundaries of the sample and alter the behavior of the crack. Another example is shock waves. The deformation caused by the shock waves generates defect bands. If the sample is too small, the defect bands are folded back through periodic boundary conditions and may mimic an amorphous state. Finite-element calculations with continuum models can often be applied in large scale simulations. But it is necessary to compare the results to atomistic simulations to find out whether they are correct.

At our institute we have developed the classical molecular dynamics simulation program IMD (**I**TAP **M**olecular **D**ynamics). The details of the basic implementation have been described in Ref. 1. Here we report on new features, especially online visualization and metacomputing together with new performance numbers for IMD.

2. Description of IMD

IMD is a software package designed for classical molecular dynamics (MD) simulations in two and three dimensions. It can be used for equilibrium and nonequilibrium simulations like heating, cooling or deformation of the sample are also possible.

IMD is a scalable program designed to get a good performance on massively-parallel supercomputers. It uses the message-passing paradigm to transfer data from one processor to another. For MD simulations with short-ranged interactions, the simulation box is subdivided into cells with the size of the interaction radius. These cells are distributed on the processors. Each processor gets a rectangular block of cells. After each simulation step, the coordinates, velocities and forces of the atoms in the surface cells of neighboring blocks have to be exchanged. This would lead to a huge number of small messages and to a large addressing overhead. Therefore the messages are collected into buffers, one for each coordinate direction, and then sent in one data exchange. Details of the implementation are given in Ref. 1. In this paper, it has been shown that IMD scales very efficiently with the number of processors if the number of atoms per cells is kept constant, thus permitting huge simulations. It also scales reasonably well if the total number of atoms is kept constant and the number of processors is increased, thus permitting very long runs which are interesting for slow cooling simulations, for example.

Originally IMD has been written on an Intel Paragon, but has been ported to the Cray T3E and the Hitachi SR2201. For communication, the standard message passing interface MPI is used. Routines like *shmem* have been avoided because this would reduce the portability of the program. IMD has also run successfully on workstation clusters. A serial version of IMD (without using MPI) is available for UNIX workstations. It is actively supported on Compaq (DEC) Alpha, Silicon Graphics, and PCs running Linux. Although there exist versions for Cray C94 and NEC SX4, we cannot recommend running IMD on parallel vector computers. The performance is low due to short vector lengths. As IMD is written in C and uses MPI, it should be no problem to port it to any UNIX workstation or massively parallel supercomputer.

IMD allows the particles to interact via pair, three-body and embedded atom (EAM[3]) potentials. Simulations of liquid crystals with anisotropic potentials are also possible. The particles can be of an arbitrary number of types with different masses. The interaction potentials are read from tables and are therefore not limited to analytical expressions. The simulations can be carried out in microcanonical (NVE), and canonical (NVT, NPT, and NσT[a]) ensembles. The Nosé–Hoover thermostat is used for the canonical simulations. In addition to MD simulations it is possible to optimize the potential energy of a sample with the microconvergence method.[4] Mechanical boundary conditions allow the sample to be pulled apart or sheared at a constant rate or at certain time intervals.

IMD contains routines to compute self-correlation functions and to generate several crystal or quasicrystal structures. In addition there are special setups for fracture simulations to damp out shock waves emitted from a crack, and in the shock wave simulations, to initiate a shock wave.

[a] σ denotes uniaxial stress.

We have grouped a number of utilities to compute static radial distribution functions, diffraction diagrams, and pictures of the particles or the defects around IMD.

2.1. *Metacomputing*

Even a massively parallel supercomputer may be too small for a certain application. The only possible way to solve this problem is to cluster a number of supercomputers worldwide. This is called heroic[5] metacomputing.[b]

IMD uses the PACX[c] library for metacomputing, which is developed at the Supercomputer Center (HLRS) in Stuttgart.[6] Since this is an almost complete implementation of MPI, only minimal changes should be required for an existing massively-parallel program.

On each of the supercomputers, two processors (PE) are reserved for communication, one for sending and one for receiving. Figure 1 shows the communication between two hosts. The communication within one supercomputer is unchanged. But if a PE on one node wants to communicate with a PE on the other node, a two-step process is initiated: first the message is sent locally to the communication PE, then it is sent to the communication PE on the other node and transmitted again over the local network to the destination PE. The communication with PACX was originally limited to two participants. Meanwhile it has been extended to an arbitrary number of computers.

IMD has been written for massively-parallel supercomputers where the communication between the PE is very fast and and the communication time is short compared to the time required for a time step. For metacomputing this is no longer the case. The latency time for a message between two partitions on the same T3E

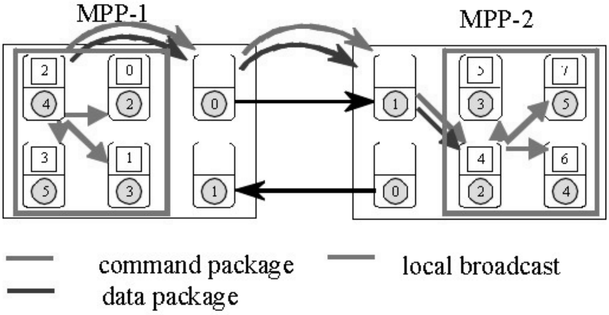

Fig. 1. Communication between two supercomputers. The labels in the squares and circles are the global and local processor addresses, respectively. The straight arrows indicate how the communication processors are connected.

[b] The opposite is dotcommish metacomputing: more or less an intelligent administration of resources but no research.

[c] PACX: **PA**rallel **C**omputer e**X**tension.

is 3.5 ms, between Stuttgart and Manchester it was 45 ms, and between Stuttgart and Pittsburgh it was 75 ms at the SC '98![8] Improvements of the PACX library have reduced the time between Stuttgart and Manchester to 30 ms at the SC '99.[8] The time between Stuttgart and Pittsburgh, however, is still 80 ms. The difference may be caused by the quality of the intercontinental connection.

The solution of the commmunication problem is called latency hiding. Computation and communication are carried out at the same time. The main loop is broken up into parts. The processor updates the interior parts of its block first. This can be done without communication. Meanwhile the data from the neighboring surfaces cells are transmitted. The processor continues the computation with the surface cells after the communication has been completed.

2.2. *Online visualization*

Previously, an MD simulation was carried out, data were collected during the run and stored for analysis which was carried out *after* the simulation has been finished. With the large massively-parallel supercomputers, such a scheme may no longer be feasible since in many cases the amount of data is too large to be stored. A solution of this problem is to visualize the data online. More advanced schemes permit steering of the simulation online.

The simulation is started interactively or in batch as usual. With IMD it is possible at any time during the simulation to open a connection to the supercomputer and to get pictures or histograms by starting VolIMD[7] on the workstation.

The communication scheme implemented in IMD works in the following way: the central part of a MD simulation is the main loop for integration of the equations of motions. At a certain time interval a function, `check_sockets`, is called which connects to a specified TCP/IP port at the workstation and looks to see whether certain flags have been set. If this is the case, the simulation program reacts as required and sends a picture or histogram of the kinetic or potential energy distribution to the workstation. The advantage of this scheme is that it is possible even if a firewall exists between the workstation and the supercomputer.

The output produced by the online visualization routines are raw data written in binary format. Tools are available to convert the binary data into ppm-files. In two dimensions, ppm-files can be written directly, but the possibility to alter the pictures afterwards is lost.

Histograms of the kinetic energy are useful in visualizing crack tips or shock waves, for example, whereas histograms of the potential energy are sensitive to interface atoms or stacking faults. Furthermore, it is possible to store reference potential energies at the beginning of the simulation and to subtract them later. This is helpful for complicated structures where the potential energy depends on the local atomic environment.

3. Demonstration Runs

As noted earlier, we have designed IMD as a scalable program for massively parallel supercomputers. Since we are dealing with short-range interactions, the program should scale linearly with the number of particles and also with the number of processors. These properties have to be tested in demonstration runs. Detailed performance numbers have been reported already in Ref. 1. The program has been tested for the two cases where the number of particles per processor is kept constant and where the number of particles is kept constant. The first test leads to very large systems, the second permits very long simulation times. In the first case an efficiency of 78.9% could still be reached with 18 966 528 particles on 512 PE, whereas the efficiency dropped to 48.5% for 296 352 particles on 512 PE in the second case. This is due to the increasing number of surface cells in the second case which leads to an increasing communication overhead. Although the efficiency numbers seem to be low, they are still impressive, especially in the second case since we can reduce the simulation time by a factor of 248 when compared to one processor. The culmination of the tests reported in Ref. 1 was a world record with 1 213 857 792 particles which took 30 min for 10 steps. This record and a number of older benchmark data are listed in Table 1. The new run with 5 180 116 000 particles took 40 min for 5 steps. For comparison, a single simulation timestep corresponds to a real time interval of the order of femtoseconds.

In the last two years, the memory of the machines has increased considerably. Recently we had the opportunity to set up a new peak performance on the T3E-1200

Table 1. Performance numbers. The third column contains the number of atoms, the fourth the number of processors, the sixth column the time per time step and the seventh the simulation time per interaction pair. We carried out a standard MD benchmark simulation for an fcc crystal with a Lennard–Jones potential and a cutoff radius of $r_c = 2.5\sigma$. The first six lines contain results with IMD, the next three are by other groups, the last line is a metacomputing simulation. There was an error in the third line which has been corrected here.

Groups	Machine	N	N_P	t_{step} s	$t_{\text{pair}}^{\text{one}}$ μs
Stadler[1]	Paragon	2 370 816	64	22.552	6.78
Stadler[1]	SR2201	296 352	8	3.675	1.11
Stadler[1]	T3E-600-64	1 213 857 792	512	159.8	1.46
Roth, this work	T3E-1200-262	1 295 029 000	128	388.8	0.84
Roth, this work	T3E-1200-262	2 590 058 000	256	389.9	0.84
Roth, this work	T3E-1200-262	5 180 116 000	512	387.9	0.83
Lohmdahl *et al.*[10]	T3D	75 000 000	128	46.9	1.46
Plimpton[10]	Paragon	100 000 000	3680	3.5	2.34
Lohmdahl *et al.*[10]	CM-5	600 000 000	1024	242	7.51
Müller *et al.*[11]	2 T3E	1 399 440 000			

at the NIC in Jülich with 262 GB memory. The number of nodes is the same as on the T3E-600 at the HLRS in Stuttgart, but each processor is twice as fast and has four times as much memory. Only one fourth of the machine was necessary to break the old record, since the memory required for the program did not grow and a larger share was available for the data. If we compare the runs with 512 CPUs by Stadler and Roth (Table 1) we find that the gain in performance is much larger than we would expect from the faster clock cycle (300 MHz and 600 MHz). If only one part of the machine is used, we find the performance as expected. The reason for this behavior is not yet clear. It may be due to interference between our job and others if only a part of the supercomputer is available.

Table 1 further contains the first record set up by a metacomputing demonstration run. Future prospects are to run MD simulations on massively parallel computers from different vendors. Although this is possible in principle with the current implementation of IMD, it has not been tried out yet, since it requires some improvement of the distribution of the work load on different computers.

Acknowledgments

We thank Jörg Stadler and the IMD team at the Institut für Theoretische und Angewandte Physik for their help. Besides the authors, the team currently includes Erik Bitzek, Matthias Hess, Rebecca Janisch, Christopher Kohler, Christoph Rudhart, Gunther Schaaf, and Joachim Stelzer.

This work has been supported in part by the Deutsche Forschungsgemeinschaft through the DFG Collaboratory Research Center “SFB” 382 and the DFG Priority Program “Quasikristalle”.

References

1. J. Stadler, R. Mikulla, and H.-R. Trebin, *Int. J. Mod. Phys. C* **8**, 1131 (1997).
2. IMD: http://www.itap.physik.uni-stuttgart.de/~imd/index.html.
3. M. S. Daw and M. T. Baskes, *Phys. Rev. B* **29**, 6444 (1984).
4. J. R. Beeler, *Radiation Effects Computer Simulations* (North Holland, Amsterdam, 1983).
5. S. Sanielevici, 2nd Metacomputing Workshop, Stuttgart 1999, http://www.hlrs.de/news/events/1999/metacomputing.html
6. PACX: http://www.hlrs.de/structure/organisation/par/projects/pacx-mpi/.
7. VolIMD is a special version of VolRend (see Ref. 2). VolRend homepage: http://www.uni-stuttgart.de/RUSuser/vis/People/roland/volrend/volrend.html.
8. The HLRS at the SC ’98: http://www.hlrs.de/news/events/1998/sc98.
9. The HLRS at the SC ’99: http://www.hlrs.de/news/events/1999/sc99.
10. D. M. Beazley, P. S. Lomdahl, N. Gronbech-Jensen, R. Giles, and P. Tamayo, *Annual Reviews in Computational Physics* Vol. 3 (World Scientific, Singapore, 1995).
11. M. Müller, *BI, RUS Stuttgart* **11/12**, 15 (1997).