Molecular Simulation, 1990, Vol. 5, pp. 133–165
Reprints available directly from the publisher
Photocopying permitted by license only

MOLECULAR DYNAMICS SIMULATION ON A PARALLEL COMPUTER

H. HELLER, H. GRUBMÜLLER and K. SCHULTEN*

Beckman Institute and Department of Physics, University of Illinois, 405 N. Mathews Ave., Urbana, IL 61801 U.S.A.

(*Received March 1989, accepted October 1989*)

For the purpose of molecular dynamics simulations of large biopolymers we have built a parallel computer with a systolic loop architecture, based on Transputers as computational units, and have programmed it in occam II. The computational nodes of the computer are linked together in a systolic ring. The program based on this topology for large biopolymers increases its computational throughput nearly linearly with the number of computational nodes. The program developed is closely related to the simulation programs CHARMM and XPLOR, the input files required (force field, protein structure file, coordinates) and output files generated (sets of atomic coordinates representing dynamic trajectories and energies) are compatible with the corresponding files of these programs. Benchmark results of simulations of biopolymers comprising 66, 568, 3 634, 5 797 and 12 637 atoms are compared with XPLOR simulations on conventional computers (Cray, Convex, Vax). These results demonstrate that the software and hardware developed provide extremely cost effective biopolymer simulations. We present also a simulation (equilibrium of X-ray structure) of the complete photosynthetic reaction center of *Rhodopseudomonas viridis* (12 637 atoms). The simulation accounts for the Coulomb forces exactly, i.e. no cut-off had been assumed.

KEY WORDS: Molecular dynamics simulation, parallel computers, parallel programming, Transputer, photosynthetic reaction center.

1. INTRODUCTION

A major concern of molecular biology is to understand the structure-function relationship of biological polymers, mainly proteins and nucleic acids. For a long time it had been tacitly assumed that the function of a biopolymer can be revealed from its static structure, i.e. from a precise knowledge of the equilibrium positions of its atoms together with a knowledge of typical atomic charges and chemical properties like hydrogen bonding. However, during the past decade it has been realized that further properties, which are not evident from the characteristics of the separate constituents of biopolymers, are required to understand function. Such properties are, for example, thermal mobilities of atoms, activated motions of constituent groups, local electric fields and dielectric relaxation.

The properties mentioned are often very difficult to measure experimentally even for small subsections of biopolymers, let alone for the whole polymer. It appears that the required information can be obtained only by computer simulations of biopolymers. Currently, many groups are developing a software basis in order to allow an increasingly faithful representation of biopolymers by computer programs. These programs are likely to contribute to biology and biotechnology beyond the scope of

*To whom correspondence should be sent.

molecular dynamics simulations, namely the force evaluation and the integration step[1], to a parallel computer and to employ existing simulation programs and graphics packages for an analysis of the simulated trajectories. To implement this suggestion a program on a parallel computer needs to interface with some existing simulation software using standard input and output files. The program described below provides such an interface specifically for the simulation programs CHARMM [6] and XPLOR [7, 8].

The parallel computer built and programmed by us is based on Transputers as computational units. The Transputer is a 32 bit processor with a 64 bit floating point coprocessor integrated on a single chip. We have chosen this chip for our parallel computer for reasons detailed further below. One advantage of the choice of the Transputer is that parallel computers, comparable to the one developed by us, can be obtained from commercial manufacturers. Therefore, molecular biologists not willing to build computer hardware, which we assume is the majority, can still use our simulation program. The reason why we decided to build our own computer rather than use a commercial machine is the following: We wanted to choose an optimal computer design in order to achieve for a minimum cost a rate of computation comparable to that of the best currently available supercomputers.

Our program has been written in occam II [9, 10, 11, 12], the language around which the Transputer had been designed. The language facilitates distribution of computational processes among Transputers as well as communication among these processes. In the initial phase of the research described here, occam II had been the only programming language for the Transputer. However, since that time more conventional languages, e.g. FORTRAN and C, have been ported to the Transputer. Molecular biologists who prefer these languages over a new language might want to incorporate the programming strategies presented below in these conventional languages. Such approach would actually allow to include elements of programs, written for sequential machines, into the program for a parallel Transputer-based machine.

The software development environment chosen by us has been the Transputer Development System (TDS), also described below, which runs on IBM personal computers. Recently, familiar operating systems, e.g. UNIX-like systems, have also been adapted to Transputer workstations. We point this possibility out, since the aim of this paper is not to propagate a particular parallel computer and a particular concurrent algorithm but rather to provide a convincing example which demonstrates the feasibility and the cost effectiveness of molecular dynamics simulations on parallel computers.

It must be pointed out that we succeeded in making molecular dynamics simulations more affordable because we had ready access to supercomputers such that programming strategies for large scale molecular dynamics simulations became familiar to us. Our experience that the use of supercomputers does not always lead to bigger appetite for expensive computational equipment, but rather can have the opposite effect, will not be an isolated one. Future development of parallel approaches to molecular dynamics simulations will require numerical experiments on conventional supercomputers.

[1]The latter step does not require much time; however, it is so closely linked to the force evaluation and, therefore, we do not want to separate the two.

where $\vec{r}_i$ denotes the position of the i-th atom. Here we have used the notation $\nabla_i = \partial/\partial\vec{r}_i$. The function E

$$E = E_B + E_\theta + E_\phi + E_{El} + E_{vdW} + E_H + E_I \tag{2}$$

defines the total energy of the molecule. It is comprised of several contributions which correspond to the different types of forces acting in the molecule. The first contribution describes the high frequency vibrations along covalent bonds, the second contribution the bending vibrations between two adjacent bonds and the third contribution the torsional motions around bonds. The fourth contribution describes electrostatic interactions between partial atomic charges, the charges being centered at the positions of the atomic nuclei. The next term, E_{vdW}, accounts for the van der Waals-interactions between non-bonded atoms in the molecule, E_H stands for the energy of hydrogen bonds, and the last term describes so-called improper motions of one atom relative to a plane described by three other atoms. Various research groups have developed functional representations and corresponding force constants which attempt to faithfully represent atomic interactions and dynamic properties of biomolecules [15]. The program which we have developed is based on the energy representation of CHARMM [6]. Actually, our program can read a file of force parameters which has a format identical to that of XPLOR [7, 8], a simulation program closely related to CHARMM. As a result, any adaptation of force constants suggested in the framework of CHARMM or XPLOR can be transferred to our program.

Due to the intermediate state of development of our program there exist still a few limitations in regard to compatibility with CHARMM and XPLOR: Our program cannot account for hydrogen bonding directly, except by reparameterizing the energy contributions other than those in E_H of participating atoms. Our program also does not include a special representation of water. Further differences will be mentioned in this Section.

The integration method of the Newtonian equations of motion employed by our program is the Verlet algorithm [16]. This method determines the positions $\vec{r}_i(t + \Delta t)$, of atoms i at the instant $t + \Delta t$ according to the formula

$$\vec{r}_i(t + \Delta t) = 2\vec{r}_i(t) - \vec{r}_i(t - \Delta t) + \vec{F}_i(t)\,(\Delta t)^2/m_i \tag{3}$$

where $\vec{F}_i(t)$ stands for the sum of all forces acting on the i-th atom at time t, i.e.

$$\vec{F}_i(t) = -\nabla_i E(\vec{r}_1(t), \vec{r}_2(t), \ldots \vec{r}_N(t)) \tag{4}$$

While integrating the Newtonian equations of motion computer time is spent mainly on evaluation of the two-particle interactions, i.e. of interactions connected with the Coulomb potential E_{El} and with the van der Waals energy E_{vdW}. The programs CHARMM and XPLOR avoid the prohibitive computational effort of an exact evaluation by allowing a cut-off for these interactions; this assumes that these interactions do not contribute much to the dynamics for pairs of atoms separated beyond a certain distance. We have not introduced such a criterion into our program. Rather than providing a cut-off option we intend to introduce an option which makes it possible to evaluate the Coulomb interaction in a hierarchical way such that, according to a hierarchy of inter-particle distances, Coulomb forces are updated with different frequencies. Such algorithm has been suggested in [4]. An alternative most promising method for an efficient evaluation of Coulomb forces, the *Fast Multipole Algorithm*, has been developed by Greengard and Rokhlin [17].

next decision to be made is concerned with the way the processors should be linked together through data channels, commonly referred to as the topology of a parallel computer. In this section we will explain the topology chosen by us, namely, the so called systolic loop.

The computational task we are mainly concerned with is the evaluation of Coulomb and van der Waals forces. Since these forces, from a computational point of view, are very similar, we will consider Coulomb forces only. Actually, much of the program code deals with the remaining interactions included in Equation (2); however, the evaluation of these interactions, in case of large polymers, consumes only a small fraction of computer time. Since some of these interactions involve three or four atoms the evaluation requires that the coordinates of as many atoms are simultaneously available to a processor. In our approach which involves a fixed mapping of processors to the primary sequence of the biopolymer (see below) this requirement is not problematic as the mapping can be chosen as to keep atoms engaged in multi-atom interactions on one processor[3].

The Coulomb forces, which describe the electrostatic interactions in a homogeneous dielectric environment[4], depend on the charges q_i and q_j of atomic pairs (i, j) and on the corresponding vector $\vec{r}_{ij} = \vec{r}_i - \vec{r}_j$ joining the atoms at positions $\vec{r}_i$ and $\vec{r}_j$. The force between atoms i and j acting on atom i is

$$\vec{F}_{ij} = \frac{q_i q_j \vec{r}_{ij}}{4\pi\varepsilon r_{ij}^3} \cdot \tag{5}$$

The force between atoms i and j acting on atom j is

$$\vec{F}_{ji} = -\vec{F}_{ij} \cdot \tag{6}$$

On a given atom the Coulomb forces of *all* other atoms act. It is, therefore, necessary to determine $\vec{F}_{ij}$ for all pairs of atoms. There are $N(N-1)/2$ pairs for a total number N of atoms. For larger biopolymers the resulting number can be extremely large, leading to the need for parallel processing.

A key problem connected with concurrent evaluations of Coulomb forces is that each processor has to know the coordinates of all atoms. If each processor had enough memory to store the coordinates of all atoms, and if updating these coordinates did not consume essential processing time, the problem of computing Coulomb forces concurrently would be rather straightforward. In our approach (see below) to molecular dynamics simulations processing elements actually spend most time on the evaluation of forces and only little time on communication of data (coordinates, forces). However, storage requirements for atomic coordinates and for lists defining the bond structure can be considerable. A processor with 1 MByte of RAM can only store coordinates of about 3 000 atoms. If one would keep all coordinates with each processor, the rather small number of 3 000 atoms would be the limit for the largest biopolymer to be simulated, no matter how many processors are employed. If one could distribute, however, storage of coordinates over all processors (because of the need of evaluating pair interactions, coordinates have to be stored at any time in two processors simultaneously), for a 50-processor machine the largest biopolymer to be

[3]This mapping requires that some atoms are represented on more than one processor.

[4]If one wishes to determine electrostatic forces in an inhomogeneous dielectric environment, one needs to solve the Poisson equation. For a computational method, see [18]. We have adopted this method to describe the interior of proteins [3] and are currently implementing the method on our computer.

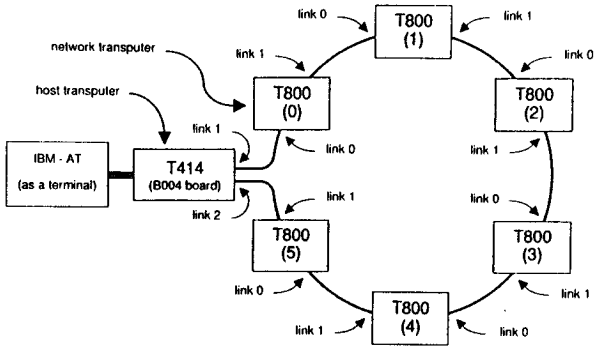

Figure 2 Ring topology adopted by us: Processors are connected to form a ring with one outlet to the host transputer. This topology had been suggested also in [19, 20].

of coordinates across other Transputers. This leads to wait states as well as impedes computations in the Transputers involved in the communication route.

The topology of the parallel computer suggested by Figure 1 is actually suitable for the simulation of particle systems with short-range interactions because the required communication routes are short. Since future versions of Transputer chips are likely to have more than four links, one will be able to map an arrangement of cells covering the simulated volume rather easily onto a set of Transputers.

The problems mentioned above are circumvented by the systolic loop topology illustrated in Figure 2, which is actually the topology adopted by us. This topology had been suggested previously as optimal for the problem at hand by Ostlund and Whiteside [19], by Hillis and Barnes [20] and by Fincham [14]. The topology connects the processors in a ring, the ring having one outlet to the host computer. Our implementation of this topology corresponds to that in Reference [20] and differs from that discussed in [19] and [14] in that the mapping between atoms and processors is fixed. The atoms are mapped onto processors (Transputers) irrespective of their position in space. In case of protein simulations one can assign Transputers to atoms in accordance with the amino acid sequence, but this is by no means necessary[5]. The atoms assigned to a specific Transputer will be referred to as the 'own' atoms of that Transputer, all other atoms are the 'external' atoms. For a discussion of the computational task of a processor we separate the Coulomb forces into two contributions

$$\vec{F}_i = \sum_{\substack{\text{'own'} \\ \text{atoms } j}} \frac{q_i q_j \vec{r}_{ij}}{4\pi\varepsilon r_{ij}^3} + \sum_{\substack{\text{'external'} \\ \text{atoms } k}} \frac{q_i q_k \vec{r}_{ik}}{4\pi\varepsilon r_{ik}^3} \tag{7}$$

$\vec{F}_i$ is the force which acts on atom i 'owned' by the processor. In order to evaluate the first contribution the processor needs to know only the coordinates of its 'own' atoms j. The second contribution, however, requires knowledge of the coordinates of all 'external' atoms k. These coordinates are passed around the ring of processors in such a way that any time coordinates pass by, a processor uses them to complete computa-

[5]A mapping of atoms following the amino acid sequence, however, is most suitable for bond interactions, e.g. bond stretch and bond angle interactions, since some of these interactions involve three or four atoms.

- an address-buffer/address-latch; these components separate address signals from data and forward them multiplexed to the memory, i.e. the components act as an interface between processor and memory.
- parity logic; this serves to detect parity errors and forwards messages about detected errors to the host transputer.
- bus bars; these devices carry the supply voltage for all components on the board; the advantage of the use of bus bars is that their high capacity provides an excellent buffering of the supply voltage.
- three LEDs (not shown in Figure 3) to indicate the current status of the node.

No ROM is employed for the computational nodes. The reason is that the Transputer feature 'boot from link' is used, and that the program is being loaded at the beginning of computations into the RAM of each node.

Since our system is designed to have 60 computational nodes[6], i.e. a rather large number, and will have to run for long periods of time to carry out simulations, one needs to be concerned with the recognition of so-called soft memory errors. One has to expect 100–1000 FITs (one FIT is a Failure in Time expected during one billion hours of operation). For a parallel computer configured with 60 MBytes of memory this error rate amounts to about one failure every few months. In order to protect calculations against such errors we store an additional parity check bit for each byte (the Transputer permits byte-wise memory access) which allows to test for single-bit errors. In case an error is detected the program can restart with the last integration step and with correct data. This makes the more complex task of error detection *and* correction, i.e. by using seven parity bits, unnecessary.

The three LEDs which are located in the front panel provide the following information:

- A red error-LED is wired to the error pin of the Transputer; it signals errors like division by zero, floating point overflow or array boundary violation.
- A yellow parity-LED indicates a parity error.
- A green busy-LED flashes with each external memory access. Therefore its intensity allows to estimate the frequency of memory accesses which in turn signals to the experienced user which kind of instructions (memory-intensive or computationally intensive) are currently being executed. Most importantly, the busy-LED signals which node is idle, a state to be avoided.

4.2 The Boards

One of the aims of the design of our parallel computer had been to make it compact. We achieved a design which allowed us to locate six Transputers with memory and necessary support chips onto a double eurocard. This was done as follows:

- We used a three layer board and chose a very high PCB density, i.e. a PCB trace width and spacing of 0.2 mm.
- We employed passive SMD components.
- We placed chips on both sides of the board: The Transputers, the parity logic as well as driver ICs were placed on the front side. The RAM-SIP modules were placed on the back. The compact design resulted in short distances which, in turn, yield well-defined signal levels and a reduced probability of error occurrence.

[6]This number, presently, is determined by the dimension of the chassis, and could be considerably larger.

In case a Transputer has to run several, e.g. 10–100, processes simultaneously, only few of these processes can hold their data in the on-chip RAM, since the Transputer must provide the workspaces for the inactive processes, too.

3. An individual Transputer is a sequential machine; if one Transputer has to run several processes, only one of them can be active at a given time, while the other processes are waiting for their time slice.
4. A most serious objection against a scheme 'one atom–one process' arises from the fact that the phenomenological force field employed in describing atomic interactions in biopolymers involves effective 3- and 4-atom interactions, e.g. describing bond angle vibrations and torsions. In order to determine such forces an atom's process would need to collect and store the coordinates of several atoms; in fact, a maximum of five atomic coordinate vectors need to be stored. This would require rather large memories for each computational node. In case of a coarse-grained programming approach with processes which monitor a few hundred atoms, the requirement on memory is much less stringent since atoms involved in multi-atom interactions often will be joined to the same process. For biopolymers this is true, in particular, when processes monitor segments of the polymer's primary sequence. In fact, such scheme leads to an increase of memory requirements for multi-atom interactions which measures only 1/15 of that for the 'one atom–one process' type of approach.

An alternative fine grained scheme of parallel computation would join processors to the terms contributing to Equation (2) in evaluating the forces $\nabla_i E$. Such approach is a generalization of schemes which join processors to atoms, since the latter schemes correspond to an accumulation of all terms arising in $\nabla_i E$, i.e. for fixed i, to a single processor. Joining processors to individual contributions of $\nabla_i E$, of which there is a number $O(N^2)$, allows an extreme distribution of computational tasks. However, such approach would result in more data flow since the evaluated contributions need to be accumulated in certain processors to yield the total forces $\vec{F}_i$. Such approach might be favourable, however, for shared-memory machines.

5.2 Transputer Development System and Folding Editor

Before we explain details of our program we like to introduce some key features of the Transputer development system [22] and its editor, the folding editor, which were used to develop and run our program.

The folding editor is key to the Transputer development system. Within this editor the compiler can be activated and so-called EXEs executed. An EXE is a program resident on a Transputer host, which, in our case, is a T414 Transputer on a modified B004 board of INMOS. The editor is named after the 'fold' which is the building block of a kind of hierarchical windowing system and which structures the source code of the program. Each part of the text is located within a fold which can be labelled by a title. A fold can be closed and, in this case, is represented on the screen by '. . . title' on a single line. Closing all subfolds one can view the global structure of the program.

Opening a fold results in a full representation of its contents on the monitor. A fold can contain other folds. Open folds can be recognized through three curly brackets which mark the beginning '{{{ title' and the end '}}}'. All editor commands, the compiler as well as EXEs are activated through function keys and usually apply only to the fold designated by the current cursor position. In keeping with the fold

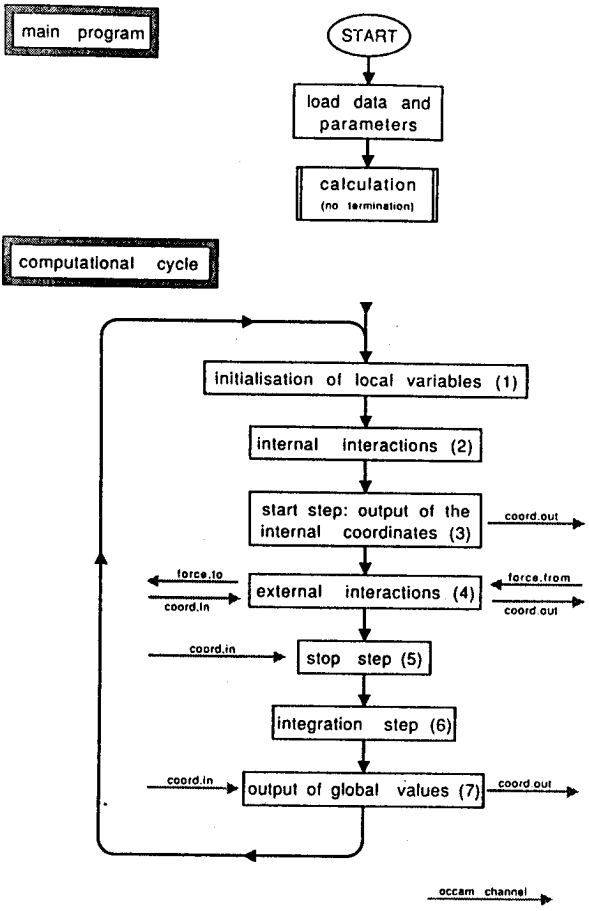

Figure 4 Flow chart of the main program. Copies of this program are running on every computational node.

Newtonian equations are integrated again for a time step Δt and a new computational cycle is started.

In a first initialization step (part 1, 2, 3 of Figure 4) each node starts computation of the forces $\vec{F}_i = \Sigma_j \vec{F}_{ij}$ acting on its 'own' atoms i beginning with the contributions of pair interactions $\vec{F}_{ij}$ originating from all of its 'own' atoms j. The node then hands the coordinates of its 'own' atoms to its next neighbour in clockwise direction and receives the coordinates of atoms 'owned' by its other neighbour. Each node keeps its 'own' coordinates in store.

The simulation program then carries out repeatedly the following steps (part 4 of Figure 4). On the basis of the coordinates just received each node evaluates pair interactions $\vec{F}_{ij}$ for the corresponding set of 'external' atoms and adds the results to the forces $\vec{F}_i$. As explained below, at this point the nodes may avoid to evaluate interactions of pairs (i, j) which previously have been evaluated for the opposite ordering of atoms, namely (j, i), by a node which 'owns' atoms j. For this purpose

$$t_M^{\text{odd}} = \underbrace{\frac{N}{M} \cdot \left(\frac{N}{M} - 1\right) \cdot 1/2 \cdot T_0}_{\text{'own' atoms}} + \underbrace{\overbrace{\frac{M-1}{2}}^{\text{other nodes}} \cdot \overbrace{\left(\frac{N}{M}\right)^2}^{\text{pairs}} \cdot T_0}_{\text{atoms of other nodes}} \tag{10}$$

Adding both terms on the right hand side yields $t_M^{\text{odd}} = t_M^{\text{ideal}}$.

In case of an even number of nodes the computation time required is

$$t_M^{\text{even}} = \underbrace{\frac{N}{M} \cdot \left(\frac{N}{M} - 1\right) \cdot 1/2 \cdot T_0}_{\text{'own' atoms}} + \underbrace{\overbrace{\frac{M}{2}}^{\text{other nodes}} \cdot \overbrace{\left(\frac{N}{M}\right)^2}^{\text{pairs}} \cdot T_0}_{\text{atoms of other nodes}} \tag{11}$$

which yields

$$t_M^{\text{even}} = \left(1 + \frac{1}{M - \frac{M}{N}}\right) \cdot t_M^{\text{ideal}} \cdot \tag{12}$$

In the limit $M \ll N$ the deviation from ideal behaviour is given by the factor $1 + \frac{1}{M}$.

This observation leads to a most interesting computational strategy: since our boards each hold six Transputers our parallel computer has an even number of nodes, say $2M$; the strategy is to assign of these nodes $2M - 1$ to the evaluation of pair interactions, and to delegate the remaining node to evaluate the forces governing hydrogen bonds; since the latter forces arise from effective four particle interactions they are best evaluated, in fact, by a single processor which can hold in its memory the coordinates of all those atoms being possibly candiates for hydrogen bond formation.

We still need to specify how the forces $\vec{F}_{ij}$ which are not evaluated by the node 'owning' atom i, but are nevertheless needed to determine the force $\vec{F}_i$, reach this node. For this purpose a communication channel for forces is established which runs in a direction opposite to that of the coordinate channel, namely counterclock-wise. If we define (node m being the sender and node n the receiver)[8]

$$A_m = \{i | i \text{ is the index for the atoms 'owned' by node } m\} \tag{13}$$

and

$$\mathbf{S}_{mn} = \{\vec{F}'_j | \vec{F}'_j = -\sum_{i \in A_m,\, i \neq j} \vec{F}_{ij}, j \in A_n\}, \tag{14}$$

then, in case of our example with six nodes, 15 such sets $\mathbf{S}_{mn}$ of interactions are generated during one computational cycle. The sets $\mathbf{S}_{mn}$ can be routed through the ring of computational nodes in the following way: After computing one of the sets each node sends its result down the ring in counterclock-wise direction. The node which needs the set fetches it and does not send it further. The routing of the sets of forces

[8] Reader please note: the indices i, j refer here to atom numbers, the indices m, n to computational nodes; the connection between the two sets of indices is determined by the assignment of atoms to computational nodes.

passing coordinates. A routing process receives incoming sets S_{mn} and passes them either (if this node is node n) through a buffer to the process which evaluates the forces $\vec{F}_j$ or (if this node is not node n) to the multiplexer which passes them further down the ring. The communication of the routing process with the process 'calculation' evaluating forces $\vec{F}_j$ is complicated by the fact that in occam II two processes cannot share data, since data may reside at different nodes. As a result the following strategy has been adopted: When a node m evaluates the sets of forces S_{mn}, $m \neq n$, it adds the results to its local force accumulators. Then it transmits the set S_{mn} to node n along the ring. When a node p ($p \neq m$, $p \neq n$) receives the set S_{mn}, its routing process recognizes that the set is not addressed to this node and passes it further. When the set reaches node n, the routing process of that node transfers it to the process 'add.buffer'. This process passes forces $\vec{F}_j \in S_{mn}$ to the process 'calculation', which adds them to the accumulators for the forces $\vec{F}_j$.

After evaluation of the total force acting on atom i, $\vec{F}_i$, is completed in each node for all of the node's atoms i, the integration step can be carried out and new sets of coordinates are available. The coordinates have to be fed now to the host Transputer and from there to the host computer for further analysis. We will discuss now how the coordinates are fetched by the host Transputer.

5.7 Processes on the Host Transputer T414

The host Transputer is part of the ring. As a result it needs to pass along coordinate packets and the sets of forces S_{mn} as well as fetch coordinates to be sent to the host computer. Furthermore, the host Transputer monitors all processors for parity errors and stops and restarts computations if necessary. The tasks described are carried out by the EXE 'controller' mentioned above. The processes and their relationships are shown schematically in Figure 5 together with the processes residing on the T800 nodes.

We first assume that a parity error did not occur and that, therefore, the parity control process did not become active, i.e. we discuss all the processes in Figure 5 except the parity control process. Most trivial is the role of the host Transputer with regard to the packets of forces handed down the ring of computational nodes; these packets are simply passed along whenever they arrive. Also the action of the host Transputer on coordinate packets is rather trivial. The master process decides if a coordinate packet needs to be sent to the host computer, i.e. to the PC AT. The actual data transfer is executed by the process 'analysis'. This process also keeps track of valid restart files and writes the trajectory data in user-definable intervals to a host file. Data concerning the various energy contributions (see Equation (2)) are also stored after each integration step in order to provide some information about the current state of the calculation to the user. Time stamps are tagged to the output data, to enable determination of the runtime required for completion of a trajectory.

We discuss now the action taken by the controller EXE in case a parity error occurs. On each node parity is checked as outlined above (Section 4). The parity error lines of all nodes are daisy-chained and connected to the event pin of the host Transputer. In case a parity error occurred anywhere in the system, the parity control process is activated and assumes control. The first action is that the process 'analysis' is informed of the event. The process 'analysis' closes down all files and assures that a valid restart file is made available. Under no circumstances will corrupted data be stored (neither in a data file nor a restart file). All other processes, i.e. 'pass-through' and

Table 2 Benchmark results of a molecular dynamics calculation on a segment of the photosynthetic reaction center (3634 atoms); shown is the time required for an integration step. The extrapolations were done on the basis of benchmarks with PTI.

VAX 11-750 *XPLOR*	*1 × T800*	*CONVEX C1* *MD [23]*	*XPLOR*	*24 × T800*	*CRAY-XMP* *XPLOR*	*50 × T800*
2 hours (estimated)	339.7s	350s	147s (extrapolated)	15.6s	7.8s (extrapolated)	7.6s (extrapolated)

for any number of nodes, as long as the number of nodes is a small fraction of the number of atoms.

Actually, computer time in molecular dynamics simulations for polymers of interesting size is spent mainly on determining the forces at each instant in time. The integration according to the Verlet algorithm (Equation (3)) takes only a small fraction of the computational effort. For this reason the benchmark tests provided here only show the time required for evaluation of the total force $\vec{F}_i(t)$, and not the time required for evaluating the remaining part of Equation (3). The benchmark tests presented vary in the degree of completion to which the total forces are determined. In one case we show results of an evaluation of solely non-bonded interactions, i.e. the Coulomb and van der Waals interaction; in another case we show results for all forces. The times needed for the computations in the different situations are given in Table 1. It should be pointed out that the computations underlying the results in Table 1 did *not* assume a cut-off of the pair-interactions, for which reason the times given in Table 1 are considerably longer than is the case for molecular dynamics calculations with cut-off.

The times shown in Table 1 need to be compared with equivalent times on conventional computers. Such times are provided in Table 2 for a VAX 11-750, a Convex C1, a Cray-XMP, a single Transputer (T800) and parallel machines with 12 and 24 Transputers (T800). The simulations compared in Table 2 involve a 3634 atom segment of the photosynthetic reaction center. As explained below the computer times needed to be extrapolated for some entries of Table 2.

We first discuss the results presented in Table 1. The time needed for simulations increases with the square of the number of atoms; the time required for evaluation of Coulomb and van der Waals forces amounts of 50–99 percent of the time needed for the evaluation of all forces, the percentage approaching 100 percent with increasing polymer size. This behaviour is understood readily. The larger the number of atoms in a polymer the more a force evaluation is dominated by Coulomb and van der Waals contributions since the respective number of terms in the total energy expression increases quadratically with polymer size.

The decrease in computer time in going from a single Transputer to 12 Transputers varies between a factor of 2.5 for the smallest polymer to a factor of 10.75 for the largest polymer. For protein segments with 3634, 5797, and 12637 atoms the rate of computation doubles in going from 12 to 24 Transputers. However, such enlargement of the computer is quite disastrous for the smaller segments simulated, i.e. those with 66 and with 568 atoms; in the first case the rate of computation actually decreases, in the second case doubling the number of processors yields only a 1.4-fold increase in computing speed.

for parts on the boards as well as the fraction of cost for chassis, power supply etc., i.e. one needs to multiply the numbers given in the Table by the number of nodes to arrive at the approximate total cost for a parallel computer.

Table 3 shows that a single node costs about \$1 000. A computer with 50 nodes, i.e. one which would equal a Cray-XMP in computational through-put would cost about \$50 000. It must be noted, of course, that the parallel computer at this stage of development lacks universality in treating different computational tasks[9]. However, this does not seem to be a problem for molecular dynamics simulations. The simulations for most investigations require computer time of such magnitude that a computer devoted solely to simulations will be welcomed by many researchers.

7. RESULTS

In this section we will present simulations of two proteins, the small protein *bovine pancreatic trypsin inhibitor*, which has been investigated many times before and serves as a test bed for our program, and a large protein complex, the photosynthetic reaction center of *Rhodopseudomonas viridis* comprised of 12 600 atoms which is probably the largest protein simulated to this day.

7.1 Simulation of Bovine Pancreatic trypsin inhibitor

The simulation of the protein *bovine pancreatic trypsin inhibitor* had been based on a structure equilibrated at 300 K [24]. An integration step of 1 fs had been adopted. In contrast to conventional procedures our simulation started with vanishing velocities.

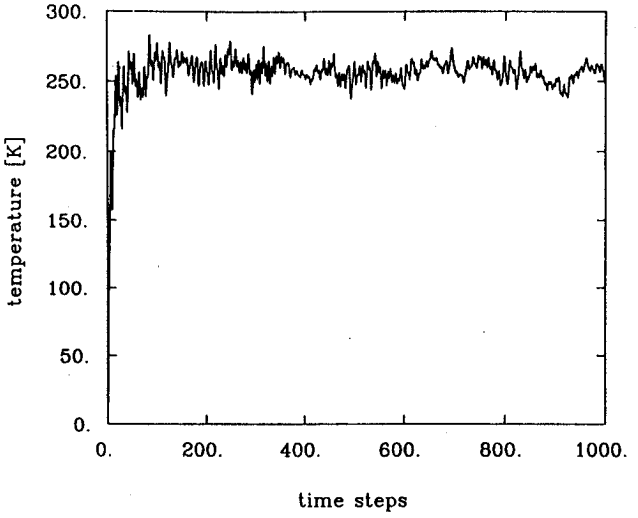

Figure 6 Time development of the temperature (defined by Equation (15)) of *Pancreatic Trypsin Inhibitor* during 1000 integration steps. The integration step size assumed was 1 fs.

[9]We have also carried out calculations on percolating systems, on the simulation of Magnetic Resonance Imaging and on Computational Neural Science on our computers. Some of these applications were programmed in Par. C.

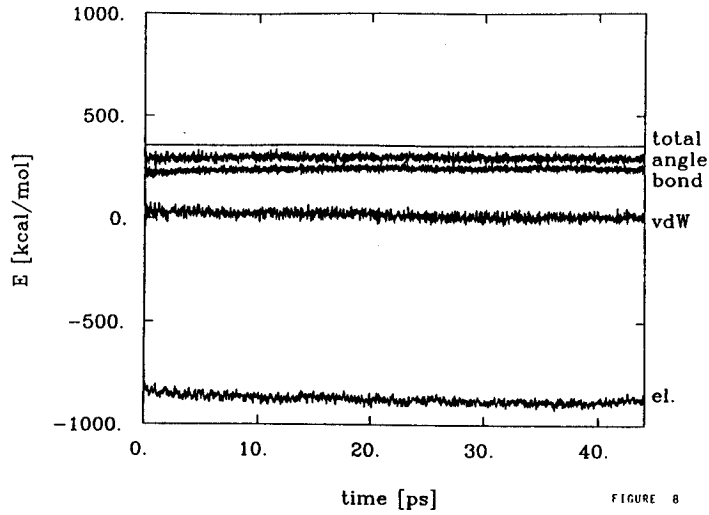

Figure 8 Behavior of the different energy contributions during the dynamics simulation of *Pancreatic Trypsin Inhibitor*; the following energy contributions are shown: total energy (total), angular energy (angle), bond stretching energy (bond), van der Waals interactions (vdW), and electrostatic energy (el.). The integration step size assumed was 1 fs.

Inserting the expression for x_{n+1} as given by Equation (16) yields the identity $x_{n+2} = x_{n-1}$ which is equivalent to time reversal symmetry.

In order to test to which extent time reversal symmetry is reproduced we have evaluated a 1 ps (*forward*) trajectory of PTI involving 1000 integration steps. At the end of this trajectory the last and second last positions were switched and the calculation continued for a further 1000 steps (*backward* trajectory). The temperature during the *forward* run, determined according to Equation (15), is presented in Figure 6. In order to facilitate comparison of temperature values of the *forward* and the *backward* trajectory we have actually plotted the temperature difference at equivalent times of the *forward* and the *backward* trajectory (Figure 7). This difference does not exceed a value of 0.0004 K demonstrating that time reversal symmetry is reproduced well. It can be concluded that round-off errors during the molecular dynamics simulation, at least for self-averaged quantities like temperature, can be safely neglected.

A further test of our algorithm is furnished by energy conservation. We have monitored the main energy contributions for *pancreatic trypsin inhibitor* during a trajectory lasting over 40 ps. Figure 8 presents the contributions of the electrostatic interactions, van der Waals interactions, angular forces, bond stretching forces and the total energy of the protein. The data presented in Figure 8 are averaged over ten successive integration steps. Figure 8 shows that the total energy remains constant over the whole time interval. The absolute energy values are in agreement with values determined in [4], except for the lowering of energy due to the cooling from 300 K to 260 K mentioned above.

Pancreatic trypsin inhibitor, being a rather small protein, is certainly not the kind of biopolymer for which the parallel computer developed by us has been intended. In fact, one aim of our development has been to specifically study the photosynthetic

Our relaxation scheme also involved some measures to control the excess energy contained in the initial structure. The respective procedures are presented in Table 5. In order to prevent the protein from becoming too hot in its kinetic energy degrees of freedom we have assumed, in the initial phase of the simulation, dissipative forces (friction) which slow down atomic motion.

When the temperature of the photosynthetic reaction center after the first 600 integration steps reached 300 K, we switched our procedure. For the following 4150 integration steps we scaled atomic velocities after each step by a common factor such that the temperature as defined through Equation (15) remained constant at 300 K. This procedure amounts to the coupling of the system to a large heat reservoir which, under realistic circumstances, is actually provided by the surrounding solvent. In order to test if equilibrium had been reached, we allowed the system to move freely for 1 950 steps, i.e. to move solely according to the integration scheme given by Equation (3). Monitoring the temperature T of the kinetic energy degrees of freedom we found, in fact, that T increased only by 16 K, i.e. the further flux of energy into the kinetic energy degrees of freedom was only minor. After this test period we resumed enforced equilibration at 300 K by rescaling of velocities. This period lasted another 350 time steps. From then on the photosynthetic reaction center was left to move freely.

The equilibrium schedule described had been devised to shorten the computational route to thermal equilibrium. A short route to equilibration is necessitated by the large size of the reaction center protein complex. The computing times for this complex make it presently impossible to apply a conventional Monte Carlo annealing scheme.

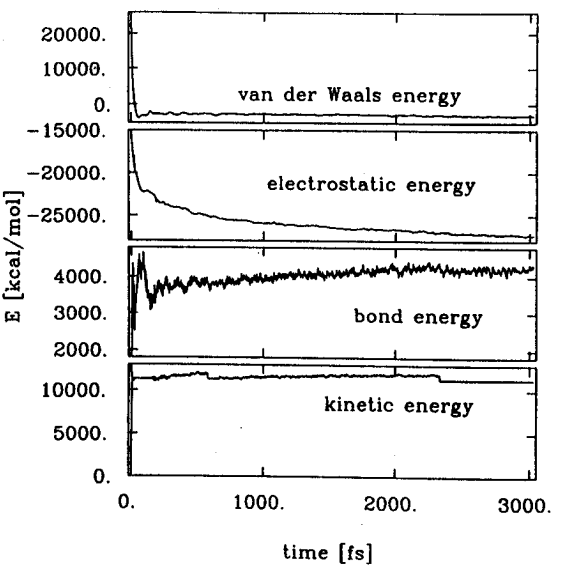

Figure 9 Time dependence of van der Waals energy, electrostatic energy, bond energy, and kinetic energy during a simulation of the photosynthetic reaction center of *Rhodopseudomonas viridis*; the simulation involved all 12 600 atoms of the photosynthetic reaction center. The details of this simulation, in particular, conditions regarding the kinetic energy degrees of freedom, are explained in the text.

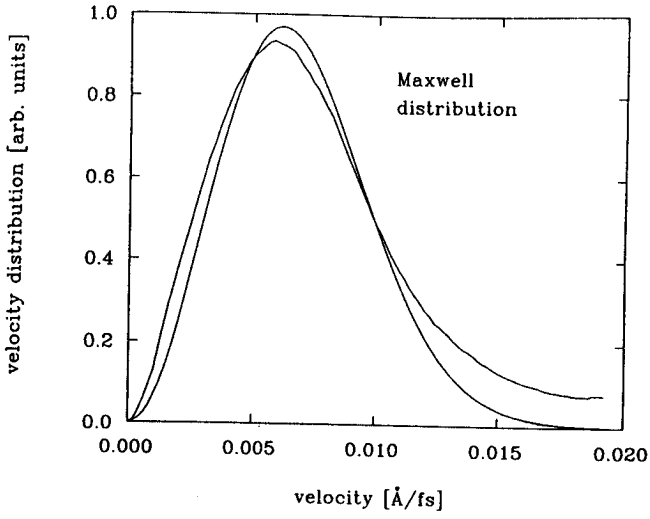

Figure 11 Comparison of the velocity distribution (at time $t = 917$ fs) resulting from a simulation of the photosynthetic reaction center and of a Maxwell distribution for a mass of 13.2 amu. This mass had been chosen to minimize the difference between the two distributions shown.

Such discrepancy is not surprising due to the fact that the atoms in the photosynthetic reaction center do not all have identical mass. The discrepancy appears to be particularly relevant for the hydrogen atoms since their mass differs most strongly from the masses of the other atoms, and since 2 344 atoms of a total of 12 637 atoms are hydrogens, i.e. over 20%. A better description of the simulated data results when one averages over the 300 K Maxwell distribution of all reaction center atoms accounting for their specific atomic masses m_j and evaluates

$$F(v) = \sum_{i=1}^{N} 4\pi \left(\frac{m_i}{2\pi k_B T}\right)^{3/2} v^2 e^{-\frac{m_i v^2}{2k_B T}} . \tag{18}$$

Figure 12 compares $F(v)$ with the simulated data. The distributions agree better than those in Figure 11, in particular in the range of higher velocities ($v > 0.015$ Å/fs) where mostly hydrogens contribute. However, $F(v)$ is narrower than the simulated distribution. The difference between the distributions in Figure 12 is due to a lack of equilibration.

Finally, we investigated the location of the secondary quinone during the relaxation of the photosynthetic reaction center. *In vivo*, i.e. for the reaction center embedded in a cellular membrane, this chromophore can leave the reaction center and, therefore, it is of interest to monitor its behaviour in this respect. Figure 13 shows the distance between the center of mass of the photosynthetic reaction center and the secondary quinone during the first 1.5 ps of the relaxation process. One can recognize a systematic drift of about 2.7 Å. It is not possible to extrapolate the behaviour seen in Figure 13 to intermediate and long times, i.e. to predict if the quinone under the conditions of our simulation would leave the reaction center. However, it might be of interest in this respect to relate an observation of the behaviour of the secondary quinone during another relaxation carried out by us. This relaxation started from an initial geometry in which sterical strain existed in some part of the phytol chain of the secondary

photosynthetic reaction center accommodates docking, penetration and undocking of the secondary quinone rather easily.

8. CONCLUSIONS

We have demonstrated in this paper that molecular dynamics calculations can be carried out in a most cost-effective manner by parallel computation. We explored, in particular, the performance of a parallel computer with a systolic ring topology and Transputers as computational nodes. For this purpose we have built such a computer and programmed it in occam II. The program's input and output files have been chosen identical to those of the well-known CHARMM and XPLOR molecular simulation programs, guaranteeing that performance can be judged and that the results of the program can be further analyzed on sequential computers by programs well-developed for this purpose. We have tested the program carrying out some representative molecular dynamics calculations on *bovine pancreatic trypsin inhibitor*. We have also demonstrated that a parallel computer with 12 and 24 nodes can simulate the dynamics of the photosynthetic reaction center, a protein complex with 12 637 atoms. No cut-off of pair interactions had been assumed in this simulation.

In closing we would like to comment on the prospects of molecular dynamics simulations on parallel computers. The merit of a parallel approach to molecular dynamics lies in its cost-effectiveness; one cannot claim at this point that the parallel approach considerably extends the range of biopolymer simulations beyond what conventional algorithms offer, except if one would invest in a massively parallel computer with 1000 nodes, say, for a price of todays supercomputers. In the latter case a rate of computation, which is twenty times higher than that of current supercomputers, would result.

Because of its cost-effectiveness the parallel approach allows a more wide-spread application of molecular dynamics simulations. For this purpose a parallel computer dedicated to biopolymer simulations should be linked to a suitable high-end graphics workstation, the latter serving for data analysis (using conventional molecular dynamics programs) and for visualization. Such simulation workstations could compute biopolymer systems of 10 000 atoms or more and allow docking of adsorbates to proteins or nucleic acids, e.g. in drug design. A most promising area where such workstations could serve a useful purpose would be in crystallographic refinement by simulated annealing. This method, suggested recently by Brünger *et al.* [26, 27] involves the simulation of a heating schedule for the molecules to be refined. These molecules are subject to restraining forces which originate from the difference between observed and calculated structure factor amplitudes. Because of the nature of the Fourier transform involved in determining the structure factor these forces are long-range multi-particle interactions for which one cannot assume cut-off approximations. Because of their multi-particle character the algorithm suggested here for two-particle interactions would need to be modified.

Acknowledgements

The authors like to express their gratitude to Zan Schulten, Markus Tesch, Andreas Windemuth, Christoph Niedermeier and Herbert Treutlein for their support which was essential to the work presented here. The authors like to thank also the Ministry

[23] A. Windemuth, "MD", August 1988, C-source of his molecular dynamics program, private communication.
[24] This structure has been given to us by A. Windemuth [4].
[25] H. Treutlein, K. Schulten, J. Deisenhofer, H. Michel, A. Brünger, and M. Karplus, "Molecular dynamics simulation of the primary processes of the photosynthetic reaction center of rps. viridis", in: J. Breton and A. Vermeglio, eds, "The photosynthetic bacterial reaction center: Structure and dynamics", pages 139–150, London, 1987. Plenum Press.
[26] A.T. Brünger, "Crystallographic refinement by simulated annealing: Application to a 2.8 Å resolution structure of aspartate aminotrasferase", *J. Mol. Biol.*, **203**, 803 (1988).
[27] A.T. Brünger, M. Karplus, and G.A. Petsko, "Crystallographic refinement by simulated annealing: Application to crambin", *Acta Cryst.*, A, in press.