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.

elucidating the structure-function relationship: they might guide the synthesis of new materials, e.g. in conjunction with genetic engineering methods, as well as predict the properties of materials, e.g. drug specificities or mechanical properties. A further long-range goal of computer simulations is to predict secondary, tertiary and quaternary structures of proteins from their primary structure (amino acid sequence).

The prospects of computer simulations in molecular biology rest on the availability of suitable computer resources. In fact, simulations until now are limited to rather small biopolymers (of a few thousand atoms) and short simulation periods (of a few nanoseconds). Furthermore, the cardinal issue of a faithful representation of biopolymers by computer simulation is closely linked to the availability of computational resources: realistic descriptions of forces acting between the constituents of biopolymers, e.g. a proper description of Coulomb forces without 'cut-off', require enormous computer time; simulations must also represent enough of the surrounding medium, e.g. lipids of biological membranes and water, in order to achieve realistic descriptions.

Our study of the photosynthetic reaction center of the bacterium *Rhodopseudomonas viridis* [1] is a case in point. We have found [2,3,4] that consideration of all atoms of the molecule and an adequate representation of electrical interactions is a prerequisite for reliable simulations. The photosynthetic reaction center is a protein complex embedded in a cellular membrane and contains 12 large prosthetic groups [5]; the whole system encompasses about 12 600 atoms. The protein complex converts the energy of the sunlight into an electric membrane potential. The two primary processes which are responsible for the high efficiency of photosynthetic energy conversion last 3 ps and 140 ps, respectively. If one wishes to include in a simulation of the primary processes some of the surrounding membrane, water and ions, one would have to simulate about 30 000 atoms over a period of a few hundred picoseconds. The necessary computations are not yet feasible except at some extreme cost, as is explained below.

Computer simulations are faced with a serious computational barrier which can be illustrated by the following estimate of the requirements on computer time: In order to determine the forces between all atoms of a protein with 12 600 atoms, i.e. of the photosynthetic reaction center, without 'cut-off', about 98 s on a Cray-XMP are needed. Since the forces have to be re-evaluated at each integration step, the size of which has to be chosen 1 fs or shorter, a simulation describing a period of 1 ns requires at least one million steps, i.e. more than 1000 days of Cray time. A 'cut-off' of pair interactions to 10 Å reduces this time to about 19 days, but one may question the soundness of such approach. The numbers illustrate a well-known point, namely, that computational requirements for molecular dynamics simulations are prohibitive and, for many problems, exceed all available means.

In this article we want to show that this situation can be improved by employing parallel computers to simulate biopolymers. For the purpose of such demonstration we have built a parallel computer with a systolic ring architecture, the design of which will be outlined. We have developed also a program for protein simulations on this computer. Computer and program achieve the same rate of computation as much larger conventional vector machines, but for a small fraction of their cost. The parallel strategy, therefore, should make the method of computer simulations accessible to many researchers, allow simulations of larger numbers of atoms as well as of more realistic (and computer time consuming) force models.

Our suggestion is to delegate only the computationally most intensive phase of

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.

A most recent review on large-scale molecular dynamics simulations of simple liquids using vector and parallel processors has been given by Rapaport [13]. This review emphasized, however, methods for handling systems of structureless particles with simple short-range interactions and, therefore, computational strategies discussed differ in most respects from the ones for biopolymers presented below. The differences are the following: (i) The force fields to be employed for biopolymers are more complex than those of condensed matter systems, i.e. there is a large family of different forces as explained briefly in Sect. 2; (ii) Furthermore, biopolymers most often are completely heterogeneous, i.e. no translational symmetry exists; (iii) The native structures of biopolymers are non-trivial, i.e. they cannot be determined by reasonable guesses and simulations, but rather need to be known beforehand from analyses of X-ray scattering data; (iv) The atoms in biopolymers have an intricate and heterogeneous pattern of chemical bonds which determines the force field and, hence, needs to be known to the simulation program. These aspects require computational approaches which differ from those taken for molecular dynamics simulations of condensed matter systems, e.g. liquids and crystals, and only few computational strategies can be shared between the approaches.

A most pertinent review on concurrent computation for molecular dynamics simulation covering algorithms for homogeneous as well as heterogeneous, e.g. biopolymer, systems has been provided by Fincham [14]. This review has influenced the work reported here, in particular, since it discussed the use of Transputers, a systolic loop architecture and respective algorithms for the evaluation of pair interactions, the latter constituting the most time-consuming task in molecular dynamics simulations.

In Section 2 we review briefly the computational aspects of protein simulation. In Section 3 we discuss the strategy of parallel computation which is dictated by the long range character of the Coulomb forces in proteins. In Section 4 we describe the parallel computer, i.e. its nodes and board design. In Section 5 we introduce our parallel program. In Section 6.1 we provide benchmark tests for actual molecular dynamics simulations comparing computational speeds with those of XPLOR running on various conventional machines producing the same output files (trajectories of various proteins). In Section 6.2 we discuss the costs of our computer system. This discussion reveals that the suggested parallel computation achieves biopolymer simulation for a much lower price than sequential computations. Finally, in Section 7, we present an application of parallel simulation, namely the relaxation of the complete structure of the photosynthetic reaction center, a system of 12 600 atoms, to an equilibrium geometry.

2. NUMERICAL TASKS IN MOLECULAR DYNAMICS SIMULATIONS

In this Section we review briefly the computational aspects of molecular dynamics simulations and discuss the relationship between our parallel algorithm and the programs CHARMM [6] and XPLOR [7, 8], to which our algorithm is closely related.

Computer simulations of biological macromolecules are based on a classical mechanical model of biomolecules. For the nuclei of the N atoms of a molecule the Newtonian equations of motion ($i = 1, 2, \ldots N$) are assumed to hold

$$m_i \ddot{\vec{r}}_i = -\nabla_i E(\vec{r}_1, \vec{r}_2, \ldots, \vec{r}_N) \tag{1}$$

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].

Calculation of van der Waals and Coulomb forces is the most time consuming task in molecular dynamics calculations. Therefore, we will concentrate below almost exclusively on the strategy to determine these forces concurrently on several Transputers.

The forces connected with the chemical bonds of biopolymers are determined much more rapidly during program execution. Because of the essentially linear arrangement of biopolymers the respective calculations can be readily ordered in a linear fashion and, therefore, a strategy for parallel computation of forces connected with chemical bonds is straightforward. Hence, we will not explain how these interactions are evaluated.

Those features of CHARMM and XPLOR which, as mentioned above, are still omitted in our current parallel program can be readily added since parallel strategies for the corresponding algorithms are known. Somewhat problematic, though, is an inclusion of the hydrogen bonding interaction since this interaction is an effective four body interaction. We are currently working on a strategy to include these interactions into our program[2]. Two further, but related differences exist between CHARMM or XPLOR and our program, which have not been mentioned yet. First, we did not include the possibility to focus a simulation on a spherical subset of protein atoms divided by a so-called 'stochastic boundary' from the remaining atoms of the protein. Second, we did not provide for the possibility to add random forces to some degrees of freedom of the Newtonian equations of motion, i.e. describe some atoms by Langevin dynamics. There is no essential difficulty connected with these features and we expect to include them into our program soon.

We would like to close this section with a brief description of the input-output requirements of our molecular dynamics program. As input the program needs a file of force parameters, a (.pdb) file of atomic coordinates in protein data bank format, and a protein structure file (.psf) with definitions of bonds, dihedral and improper angles, etc. The file formats are identical to those of CHARMM and XPLOR. As output the program delivers atomic coordinates in an internal format which is converted on the host computer into a (binary) format for analysis of trajectory properties by CHARMM or XPLOR.

3. COMPUTATIONAL STRATEGY FOR LONG RANGE PAIR INTERACTIONS

The first step in considering a computational stragegy for concurrent simulation calculations is to make a basic decision about the nature of the processing elements: should they all receive the same instructions synchronously (so called SIMD: Single Instruction Multiple Data) or should they each have a resident molecular dynamics program working on local data (so called MIMD: Multiple Instruction Multiple Data). The decision is, of course, closely tied to the hardware one wishes to employ. Since we decided to use Transputers (which are loosely coupled processors with distributed memory) as computational elements of our parallel computer we opted for a solution in which each processor executes its own molecular dynamics program. The

[2]Since the number of atoms which are possible candidates for hydrogen bonding is much smaller than the total number of atoms one can actually devote a single computational node to the task and compute hydrogen bonds in a conventional, i.e. sequential, manner (see also below).

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.

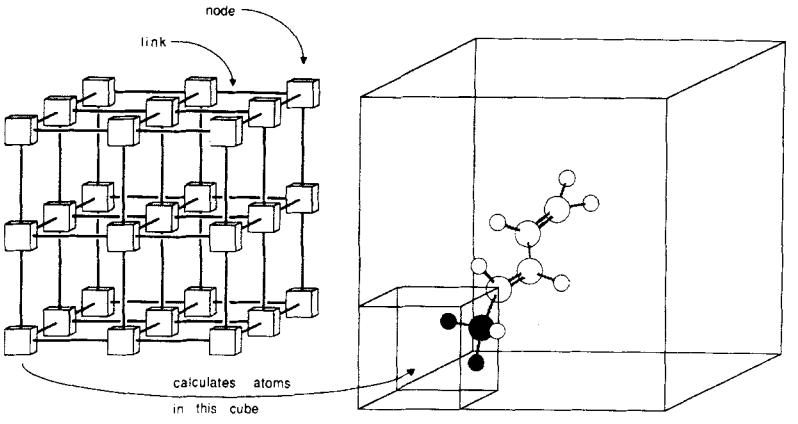

Figure 1 Cube topology: Each processor is assigned to simulate all atoms within a spatial cell, e.g. within a cube. This topology has not been adopted by us.

simulated could encompass about 75 000 atoms. Because the latter storage mode, obviously, is more favorable, one likes to adopt a computational strategy which distributes the storage of forces and coordinates as well as the actual computation over all computational nodes. As already mentioned, data transfer is not critical for molecular dynamics simulations, at least not for *large* biopolymers.

Another important consideration in devising a programming strategy for a parallel computer is to take advantage of the fact that for any pair of atomic charges q_i and q_j, the Coulomb forces $\vec{F}_{ij}$ and $\vec{F}_{ji}$ acting on q_i and on q_j, respectively, are related by Equation 6. Of course, one wishes to avoid evaluating both forces, i.e. duplicating computational effort. A programming strategy which avoids such duplication requires that forces are communicated among processors.

One possible way to link processors and computational as well as storage tasks is to assign atoms, i.e. the evaluation of the forces acting on them, to processors according to the spatial arrangement in the protein. Such assignment is referred to as a geometric mapping. An example is illustrated in Figure 1: to each processor are assigned all atoms within a spatial cell, e.g. within a cube. For our purpose the seemingly natural geometric mapping of Figure 1 has serious disadvantages:

- The geometric mapping in Figure 1 would require each processor to communicate with the neighbouring processors, i.e. with six processors. However, the Transputer currently has only four links for communication with other Transputers, i.e. Transputers cannot reproduce the connectivity required for this mapping.
- An inhomogeneous distribution of atoms over the simulated volume can lead to a poor balance of computational tasks of processors, i.e. some processors may need to monitor more atoms than others and, consequently, would make other processors 'wait' which, therefore, remain idle. This problem could be alleviated by using a flexible spatial grid which, however, would lead to some extra overhead in assigning atoms to 'their' Transputer.
- Atoms which switch between cells require additional communication between processors of neighbouring cells.
- Evaluating pair interactions between atoms separated by a large distance, i.e. between atoms in two cells separated by several other cells, requires communication

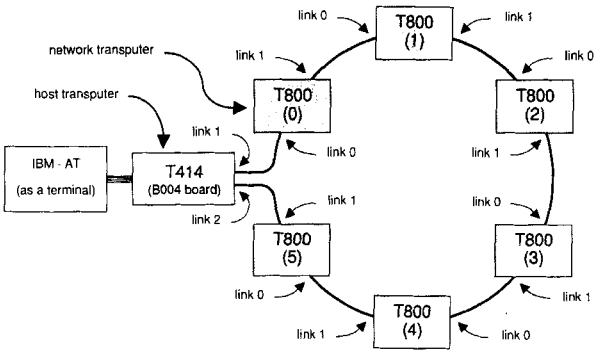

Figure 2 Ring topology adopted by us: Processors are connected to form a ring with one outlet to the host transputer. This topology had been suggested also in [19, 20].

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.

tion of the total forces $\vec{F}_i$ that act on its 'own' atoms. The ring topology also allows to avoid unnecessary evaluation of forces of pairs of charges related by Equation 6. The details will be explained further below.

4. HARDWARE

Having decided on a topology for the parallel computer, we like to describe briefly the hardware developed by us for the purpose of molecular dynamics simulations. We like to point out again that an application of our computational strategy does not require a prospective user to rebuild the same parallel computer; a user may also acquire a similar hardware available through several commercial manufacturers.

4.1 Structure of a Single Computational Node

A computational node, placed on a board with five other nodes, is shown schematically in Figure 3. The node has been constructed from the following components:

- a Transputer IMS T800G20S as the central processing unit of the node; this processor has a 32 bit data and address bus and runs on a 22.5 MHz cycle; the processor also contains, integrated on a single chip, a 64 bit floating point coprocessor.
- 1 MByte dynamic RAM (four SIP modules each with nine 100 ns 256 Kbit chips; 8 chips contain data, the 9th chip carries parity information) as local memory to store program (currently about 30 KByte) and data; the size of the memory can be readily extended to 4 MByte by using 1 Mbit DRAM chips.

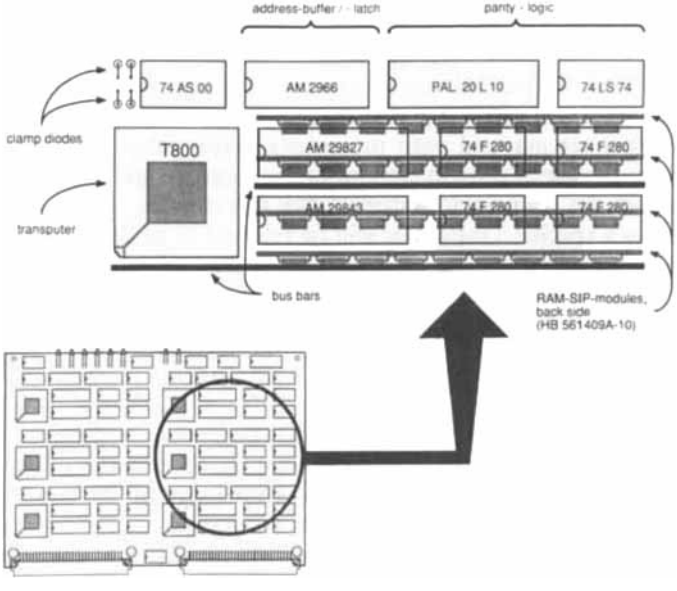

Figure 3 Schematic drawing of a transputer-board enlarging a part of the board corresponding to one computational node. Components on the board are explained in the text.

- 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.

The simulations presented below were carried out by means of these boards. We have recently developed new (six layer) boards each with six Transputers and 4 MByte RAM per Transputer.

4.3 Backplane and Chassis

The 'topology' of the parallel computer is defined through the backplane. For this purpose all links of the Transputers on the boards (see above) have been connected to the backplane. The backplane establishes the connections between the computational nodes. For that purpose the backplane, currently, has a layer with fixed connections which define the ring topology. However, by placing link switches (IMS C004 [21]) onto the backplane the topology can be reconfigured even during program execution.

The chassis (19") holds the backplane and provides ten slots for Transputer boards, each board contributing six computational nodes. 60 Transputers providing a computational through-put for molecular dynamics calculations exceeding that of a Cray XMP (see below) fit into the chassis. The backplane contains an interrupt logic which allows to separately access any of the 60 Transputers.

5. THE PROGRAM

5.1 Coarse-Grained versus Fine-Grained Parallel Program

We want to address first the question of why we have opted for a coarse-grained approach to parallel computation, i.e. the smallest computational node being a powerful processor on which a rather long serial program is being executed. One may argue that the best approach to parallel computation of polymer dynamics would be to adopt a parallelism which matches that of the physical system described and employ one computational node for each atom. Such fine-grained approach would simplify the structure of the parallel algorithm, i.e. one would not have to differentiate between 'own' and 'external' atoms (see Chapt. 3).

The terms coarse-grained and fine-grained are meant to refer here to the software, rather than the hardware, i.e. a fine-grained approach implies a program for which the longest process uninterrupted by communication of data to other processes is rather short. In fact, we envision that the fine-grained approach suggested would be realized on a set of Transputers as well, such that each Transputer accommodates through its hardware scheduler a number of processes each monitoring the dynamics of a single atom. In occam II the programmer would actually write his program as if each of the processes joined to an atom would run on a separate Transputer, and it is the on-chip logic of the Transputer which schedules these processes to run on the available computational nodes.

The seemingly natural approach of matching the parallel program process structure to atoms would lead to a run-time disadvantage. The reasons are the following:

1. Each parallel process involves a certain scheduling overhead which cannot be ignored despite the fact that the scheduling is done by the hardware scheduler of the Transputer.
2. The 4 KByte on chip RAM of the Transputer is a most valuable resource since it allows faster data access than external memory. In case a Transputer runs a single process, the on-chip RAM can hold all scalar data and access these data quickly.

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

organization of the Transputer development system we use an EXE to write the atomic coordinate file and force parameter file into a fold with a hierarchical subfold structure. This allows, in particular, ordering of data in input files and, consequently, speeding-up initial loading of the computational nodes with input data. The folding editor also permits some editing of input data without leaving the Transputer development system.

A simulation is started by calling the 'controller' EXE with the cursor placed on the fold containing the relevant molecular data.

5.3 Modular Structure of the Program

The program has three modular parts. The first module provides an interface to the protein data bank (.pdb) file format as well as to CHARMM (XPLOR) files which contain information on protein atomic coordinates and the parameterization of force fields. The second module is a controller program which handles data transfer of the main program, i.e. of the program that actually carries out the simulation. The controller program also monitors the parity flag and periodically polls the keyboard for some user action, e.g. program interruption. The third module is the main (simulation) program and runs separately on each computational node. The separation of the controller program and the main simulation program is a consequence of various organizational requirements and fits well into the Transputer development programming environment.

The controller program is executed on the host Transputer. The latter is connected to an IBM PC AT computer, but will later be connected to a high end workstation capable of molecular graphics. The controller program is written as an EXE (in the Transputer development system nomenclature). The simulation program in the same nomenclature has the form of a PROGRAM.

5.4 Loading of Data onto the Computational Nodes

After the data fold has been completed the EXE controller can start on this fold. The module first clears all parity flags and all local memories. After all nodes have received the simulation program (the same for each node) the module transfers the relevant data to the nodes. For this purpose the module needs to determine how many atoms will be 'owned' by each node. In order to evenly distribute the computational load the difference between the number of atoms 'owned' by any two nodes should be at most one atom.

5.5 Communication Channels along the Ring Architecture

We like to explain now how the computational strategy of the main program builds on the ring architecture, i.e. adopts a communication pattern compatible with this architecture. A flow chart of this program is presented in Figure 4. We will refer to this chart repeatedly in the following.

The overall structure of the simulation program is a loop over a computational cycle. A cycle starts at the moment when the program has just completed an integration step invoking the algorithm stated by Equations (3) and (4). The integration step yields a new set of coordinates. The computational cycle then uses these coordinates to determine the forces acting on the atoms. When these forces are determined the

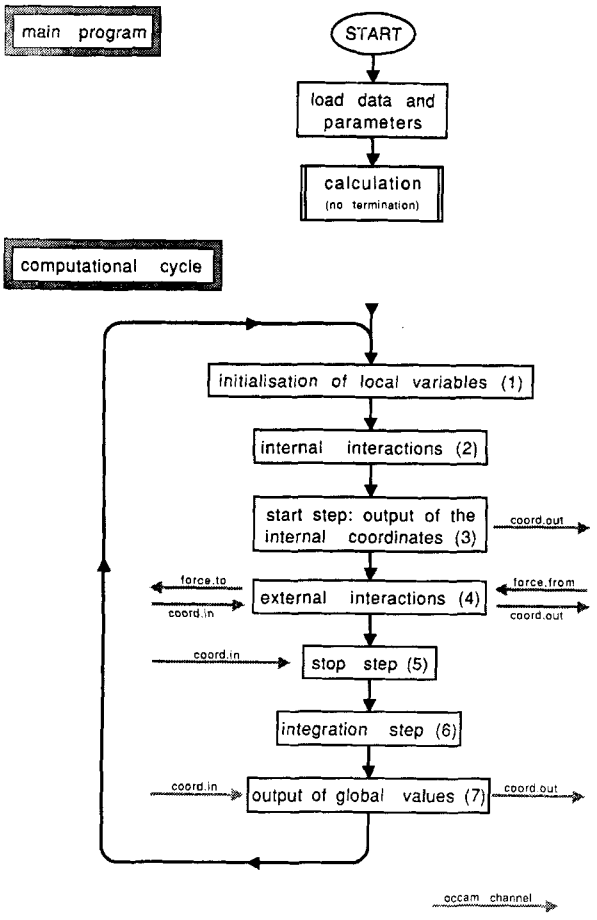

Figure 4 Flow chart of the main program. Copies of this program are running on every computational node.

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

forces, too, need to be communicated along the ring of computational nodes. After adding all possible contributions to $\vec{F}_i$ the 'external' atoms are passed by each node to its next neighbour in clock-wise direction and new 'external' coordinates are received from its next neighbour in counterclock-wise direction.

These steps are repeated until each node receives its 'own' coordinates again, at which point this loop is terminated. The strategy outlined keeps all computational nodes busy at all times except for periods when some set of nodes may have to wait for other sets of nodes to complete the computations. However, these periods can be kept very short as long as all nodes 'own' nearly the same number of atoms.

The strategy described would still be inefficient if one would not avoid that interactions for pairs (i, j) are evaluated twice, once by the node which 'owns' the atoms i and once by the node which 'owns' atoms j. The issue of avoiding redundant evaluation of pair interactions had already been addressed by Ostlund and Whiteside [19]. We will follow a slightly different strategy than these authors. In order to illustrate how redundant computations are avoided we assume a ring of six nodes labelled 0 through 5. In the initialization step each node evaluates all pair interactions among its 'own' atoms. In the further steps, node 0 then evaluates all pair interactions between its 'own' atoms and those 'owned' by nodes 5, 4, 3. Node 1 evaluates those interactions for atoms 'owned' by nodes 0, 5, 4, node 2 those of nodes 1, 0, 5. Hence, nodes 0 to 2 compute pair interactions for three sets of 'external' atoms. Nodes 3, 4 and 5 compute only pair interactions for two sets of 'external' atoms, namely in case of node 3 for those sets 'owned' by node 2 and 1, in case of node 4 for those 'owned' by node 3 and 2, and, finally, in case of node 5 for those 'owned' by node 4 and 3.

This example involving six nodes should make a generalization to an even number of nodes obvious. In case of an even number of nodes half the nodes are computing one set less than the other half. In case of an odd number of nodes the situation can be improved in that no node becomes idle. We have derived expressions which describe the decrease in computer time when M nodes compute the forces between all pairs of atoms in parallel according to our scheme. In the ideal case this decrease would correspond to a computation time

$$t_M^{\text{ideal}} = t_1/M \tag{8}$$

where t_1 is the time needed by a single processor. The latter time for N atoms is

$$t_1 = \frac{N(N-1)}{2} \cdot T_0 \tag{9}$$

where T_0 is the time required to evaluate a single interaction, which for Coulomb interactions measures about 50 μs on a Transputer.

The ideal Equation (8), in fact, can be achieved by our computation scheme when the number of nodes employed is odd. This can be shown as follows. Our scheme assignes N/M atoms to each node[7]. Each node evaluates first the pair interactions between the atoms assigned to it and then the interactions with the $M - 1$ sets of atoms assigned to other nodes, of which only half need to be evaluated. The total computation time is given by

[7]The number of atoms assigned is actually the integer closest to N/M. For large N the value is close to N/M such that we may use this ratio in our derivation.

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

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.

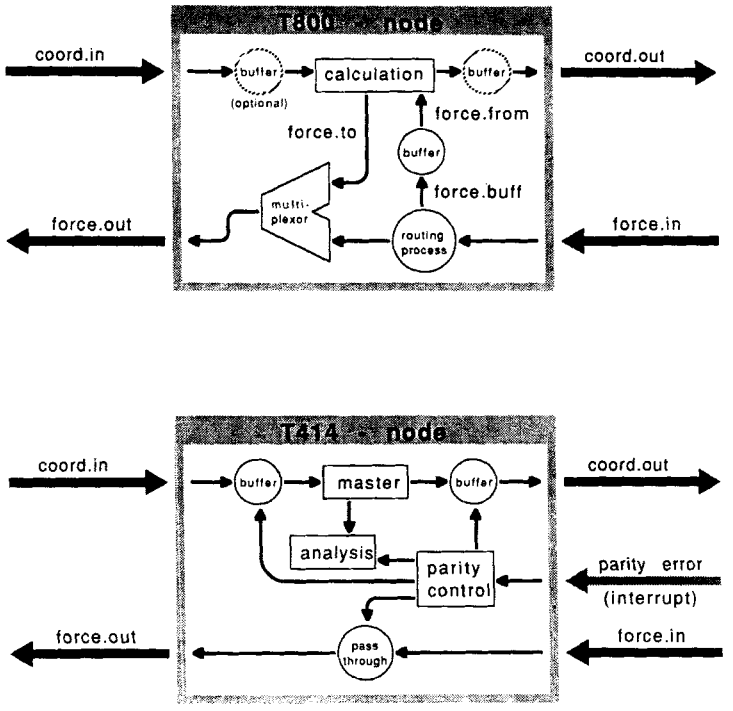

Figure 5 Process structure of the program running on the T800 nodes and of the program running on the T414 host Transputer. Shown are the processes running in parallel.

$\mathbf{S}_{mn}$ along their pathway is asynchronous to the routing of coordinate packets and evaluation of $\mathbf{S}_{mn}$. The processes need to be synchronized only as far as it is necessary that each node has to fetch the routed $\mathbf{S}_{mn}$ needed to complete evaluation of the total forces $\vec{F}_i$ for its 'own' atoms i.

5.6 Process Structure of a Single T800 Node

The computational nodes need to carry out the following processes: (i) send coordinates around the ring clock-wise; (ii) send force sets $\mathbf{S}_{mn}$ along the ring counterclockwise; (iii) evaluate force sets $\mathbf{S}_{mn}$ and total forces $\vec{F}_i$; (iv) synchronize as far as it is necessary the routing of $\mathbf{S}_{mn}$ and the evaluation of $\vec{F}_i$. The processes and their relationship are shown schematically in Figure 5.

The process in each node, which routes the coordinates and uses them to evaluate the sets of forces $\mathbf{S}_{mn}$ as well as the forces $\vec{F}_i$, is rather straightforward. In the initialization step a node uses the coordinates of its 'own' atoms to determine the sets $\mathbf{S}_{mn}$ and then passes its 'own' coordinates on. When packets of 'external' coordinates arrive they are stored and, if necessary, used to evaluate sets $\mathbf{S}_{mn}$. From this set the node determines the contribution to $\vec{F}_i$ for its 'own' atoms. The node also passes the sets $\mathbf{S}_{mn}$ to a multiplexer which feeds them into the force pathway.

The administration of the force pathway is slightly more complex, mainly because the pathway is asynchronous to the former node processes, i.e. to those processes

passing coordinates. A routing process receives incoming sets $\mathbf{S}_{mn}$ 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 $\mathbf{S}_{mn}$, $m \neq n$, it adds the results to its local force accumulators. Then it transmits the set $\mathbf{S}_{mn}$ to node n along the ring. When a node p ($p \neq m$, $p \neq n$) receives the set $\mathbf{S}_{mn}$, 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 $\vec{F}_j \in \mathbf{S}_{mn}$ 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 $\mathbf{S}_{mn}$ 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

'buffers', are forced to terminate. Since all parallel processes on the Transputer host terminate, computation can now be restarted. During restart, not only the data, but also the program of the computational nodes is reloaded, to take every possible error into account.

6. PERFORMANCE

6.1 Benchmarks

Benchmark tests to compare the performance of computers are rightly criticized as they often do not provide a basis to judge a computer's performance for special applications. Since we consider, however, a specific application, namely, molecular dynamics simulations of biopolymers, meaningful benchmark tests which compare how different computational approaches generate the same output from given input, can be provided. The output contains sets of atomic coordinates $\{\vec{r}_i(t) \mid i$ denotes atoms$\}$ at successive times $t = t_1, t_2, \ldots$ and the input consists of the CHARMM force parameters, a protein structure file and initial coordinates. The remaining uncertainty connected with such test is how performance depends on the size of the proteins simulated and on the number of computational nodes involved in the concurrent computation. With regard to the size of polymers simulated we will provide five examples, a short chain (deka-alanine, 66 atoms), a small protein (pancreatic trypsin inhibitor (PTI), 568 atoms), and three larger protein segments (parts of the photosynthetic reaction center (RC) comprising 3 634 atoms, 5 797 atoms, and the complete RC complex of 12 637 atoms). These examples allow an extrapolation to other polymer sizes. To demonstrate the dependence of computational performance on the number of nodes we provided data for a single node as well as data for 12 and 24 nodes. We will demonstrate that the scaling of the computational effort with the number of nodes follows closely the law derived in Section 5.5. This implies that from our data one can estimate rather accurately the performance of our parallel computer

Table 1 Benchmark results for a molecular dynamics simulation of Alanin, PTI (pancreatic trypsin inhibitor) and RC (photosynthetic reaction center of *Rhodopseudomonas viridis*); shown is the time required for an integration step (in seconds); *no cut off of Coulomb interactions had been assumed.* 'Optimized' refers to placing all scalar variables and the main loop of the program in the on-chip RAM of the Transputer.

	alanin, 66 atoms	*PTI, 568 atoms*	*RC, 3 634 atoms*	*RC, 5 797 atoms*	*RC, 12 637 atoms*
electrostatic and vdW-interaction, single T800	0.13	9.2	374.0		
all interactions, single T800	0.16	10.0	392.6		
all interactions, optimized, single T800	0.15	8.7	339.7		
electrostatic and vdW-interaction, optimized, 12 T800	0.03	0.8	31.3		
all interactions, optimized, 12 T800	0.06	0.9	31.6	80.8	386.9
all interactions, optimized, 24 T800	0.2	0.65	15.6	40.5	191.0

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*	*MD* *[23]*	*CONVEX C1* *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.

The small decrease of computer time in going from a single Transputer to 12 Transputers in the case of alanine and the increase of computing time in going to 24 Transputers is due to the character of the parallel program: in case of a polymer comprised of not more than 66 atoms the overhead for loops and the opening and closing of communication channels it not outweighted by the concurrency of computation. Also in case of very few atoms, the ratio of time needed for computations to time needed for communication is poor. This poor gain in computational speed, when computational nodes are increased in number, is not a serious deficit since actual computations on biopolymers usually involve a very large number of atoms with a very small ratio nodes vs. atoms.

The decrease in computer time found for the polymer with 3634 atoms in going from a single Transputer to 12 Transputers, i.e. a decrease by a factor of 10.75, is close to the theoretically best possible value (see above) of 11.08. The decrease in computing time by a factor of two for simulations of the largest proteins in going from 12 to 24 Transputers is also expected from a quantitative estimate. This finding, i.e. a behaviour close to the theoretical limit, allows one to estimate the decrease in computer time one would expect in going from a single Transputer to, say, 50 Transputers, i.e. a decrease from 339.7 to 7.6 s. These times should be compared to the computer times spent for the generation of the same output by running XPLOR on conventional machines, i.e. the times presented in Table 2. Our computations showed that our program on a single Transputer runs 20 times faster than XPLOR on a Vax 11-750 and about half as fast as XPLOR on a Convex C1. 24 Transputers show a performance nearly ten times faster than a Convex C1 and two times slower than a Cray-XMP. 50 Transputers are likely to beat the performance of XPLOR on a Cray-XMP.

At this point of our comparison the question of relative costs of computational equipment arises. This question will be addressed now.

6.2 Costs

Costs for dynamics simulations of biopolymers can become rapidly prohibitive and dominate approaches taken for most molecular dynamics studies. We like to demonstrate now that molecular dynamics calculations on Transputer-based parallel computers are extremely cost-effective. We provide in Table 3 the cost of the components needed for the building of one computational node. The prices quoted include the costs

Table 3 Costs for one node

article	*company*	*costs [DM]*
Transputer IMS T800G20S	E2000, München	1042.25
dynamic RAM, HB561409B-10	Termotrol, München	471.75
power supply type LFS-49-5	Lambda, Achern	46.25
fan, type V72AB-220VAC	Bedeke, Dinkelsbühl	4.96
diverse ICs	various	28.57
passive components	various	33.67
rack	Multimechanik, Filderstadt	3.91
bus bars	MPS, Feldkirchen-Westerham	67.22
boards	HFW, München	198.48
carriers	various	42.35
total		1939.41

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.

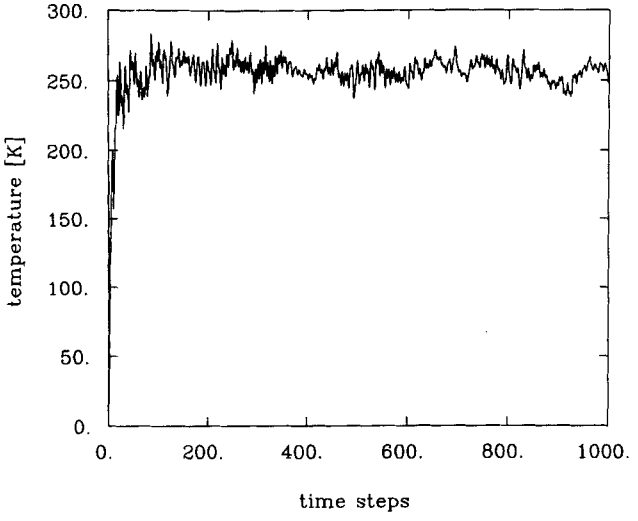

Figure 6 Time development of the temperature (defined by Equation (15)) of *Pancreatic Trypsin Inhibitor* during 1000 integration steps. The integration step size assumed was 1 fs.

[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.

Following the equipartition theorem the temperature of a system with quadratic kinetic energy degrees of freedom is given by the expression

$$T = \frac{1}{3k_B N} \sum_{i=1}^{N} m_i \langle v_i^2 \rangle \cdot \tag{15}$$

In the following we assume this as our definition of temperature. The time development of T is shown in Figure 6. T measured initially 0 K and approached rapidly a value close to 260 K. This behaviour of the temperature is due to the flow of energy from the potential energy degrees of freedom into the kinetic energy degrees of freedom. The reason why the temperature does not reach 300 K is due to the considerable cooling effect experienced by the protein when the kinetic energy had been quenched initially.

In order to test the accuracy of our simulation we used the time reversal symmetry of the classical equations of motion. The integration scheme chosen, the Verlet algorithm, obeys time reversal symmetry. This can be demonstrated by considering the one-dimensional case which gives rise to the three term recursion equation connecting x-coordinates at times t_{n-1}, t_n, t_{n+1}

$$x_{n+1} = 2x_n - x_{n-1} + \frac{F_n}{m} \cdot (\Delta t)^2 \tag{16}$$

where F_n denotes the force at position x_n. Time reversal is expressed by reversing velocities while keeping positions fixed. In the framework of the discrete Verlet recursion equation, time reversal involves a switching of old and new coordinates. The next Verlet integration step can implement time reversal by the replacements $x_{n+1} \rightarrow x_n$, $x_n \rightarrow x_{n+1}$ and leads to the new position

$$x_{n+2} = 2x_n - x_{n+1} + \frac{F_n}{m} \cdot (\Delta t)^2 \tag{17}$$

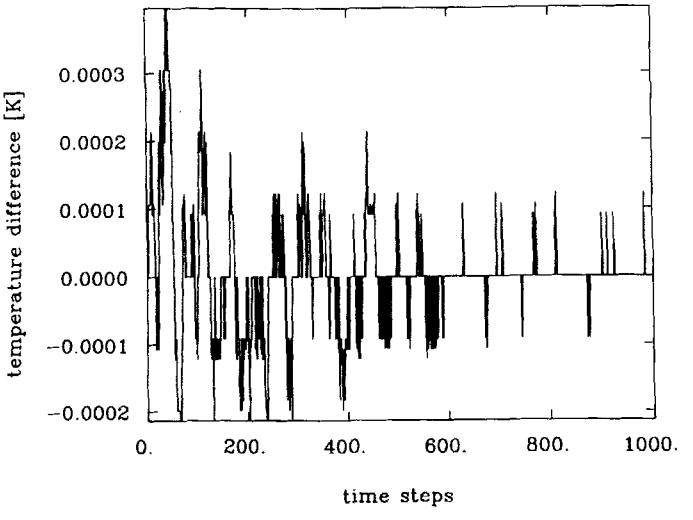

Figure 7 Temperature difference between corresponding points of a *forward* and *backward* trajectory of the simulation represented in Figure 6. The differences are due to numerical round-off errors. The integration step size assumed was 1 fs.

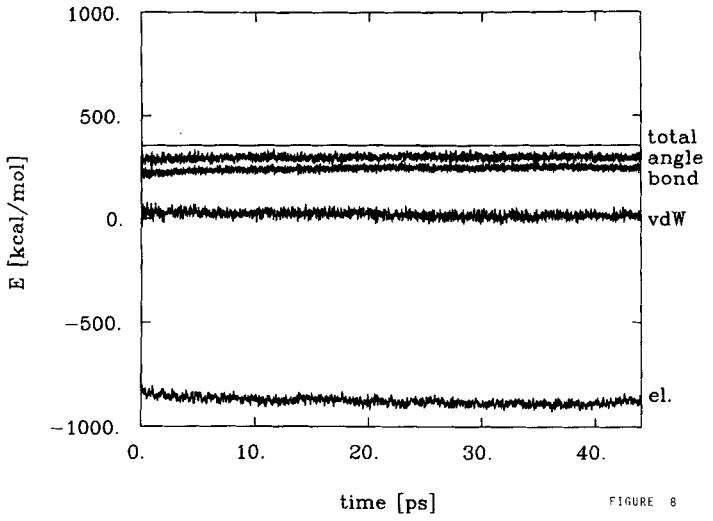

Figure 8 Behavior of the different energy contributions during the dynamics simulation of *Pancreatic Trypsin Inhibitor*; the following energy contributions are shown: total energy (total), angular energy (angle), bond stretching energy (bond), van der Waals interactions (vdW), and electrostatic energy (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

Table 4 Integration step sizes adopted during the equilibration

time [fs]	*0–35*	*35–205*	*205–*
integration step number	1–3499	3500–5199	5200–
integration step size [fs]	0.01	0.1	0.25

reaction center, a very large protein complex with 12 637 atoms. A simulation of this protein will be presented now.

7.2 Photosynthetic Reaction Center of Rhodopseudomonas viridis

Starting point of our simulation of the photosynthetic reaction center of *Rhodopseudomonas viridis* was the X-ray structure of Deisenhofer and Michel [5]. The structure was completed by us placing hydrogen atoms in a conventional way [25]. The resulting coordinates were chosen as a starting set for a molecular dynamics relaxation.

X-ray structures are obtained by fitting a given amino acid primary sequence into the electron density, the latter being directly derived from the scattering data. The fitting procedure is usually carried out assuming equilibrium values for the bond lengths and bond angles of the amino acid side groups, altering solely the dihedral angles. In a second step often a limited, i.e. local, energy optimization is carried out. The resulting structure has low energy in the bond lengths and bond angles, but in most cases less than optimal van der Waals (sterical) and Coulomb energy. When the atoms of such a structure are free to move in a molecular dynamics simulation one expects energy to flow from sterical and electrostatic interactions into bond interactions and into kinetic energy. Such energy flow, in fact, is born out by our simulation of the photosynthetic reaction center.

It should be emphasized that we have not introduced in our calculations an artificial cut-off of pair interactions. This yields a faithful representation of biopolymers, but implies very long computation times. The time required for a single step for an integration of the equation of motion of the complete photosynthetic reaction center, i.e. one application of the Verlet formula Equation (3), on a parallel computer with 24 Transputers was 191 s.

For a simulation of a relaxation process of the photosynthetic reaction center we have adopted a scheme which is described by Table 4 and 5. Table 4 lists the integration step sizes adopted. In a first period, during which strong forces arise because of sterical strain in the initial structure, we have chosen a small integration step size of 0.01 fs. This period lasted 3 500 integration steps. In the period extending from step 3 500 to step 5 200 we chose a longer step size, namely 0.1 fs. All later integration steps had a size of 0.25 fs.

Table 5 Equilibration of the photosynthetic reaction center. In the entries of the Table below 'friction' refers to a rescaling of velocities by a factor 0.9999 after each integration step, '$T = 300$ K' refers to a rescaling of velocities as to keep the temperature defined through Equation 15 constant; 'free motion' refers to a solution of the Verlet equation (Equation 3).

time [fs]	*0–6*	*6–160*	*160–580*	*580–668*	*668–*
integration step number	0–599	600–4749	4750–6699	6700–7049	7050–
method applied	friction	$T = 300$ K	free motion	$T = 300$ K	free motion

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.

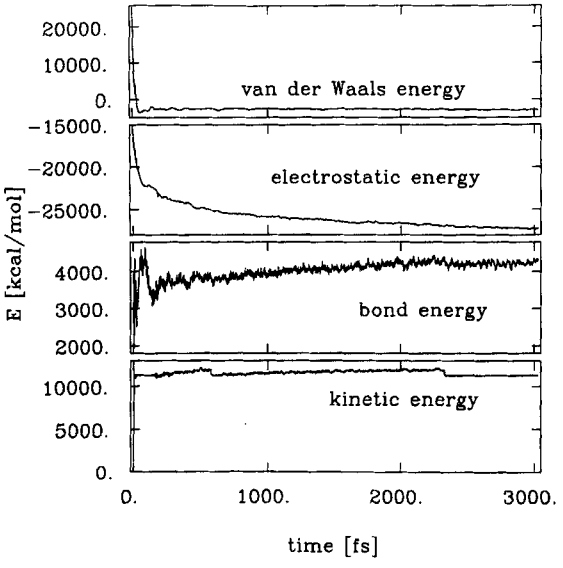

Figure 9 Time dependence of van der Waals energy, electrostatic energy, bond energy, and kinetic energy during a simulation of the photosynthetic reaction center of *Rhodopseudomonas viridis*; the simulation involved all 12 600 atoms of the photosynthetic reaction center. The details of this simulation,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

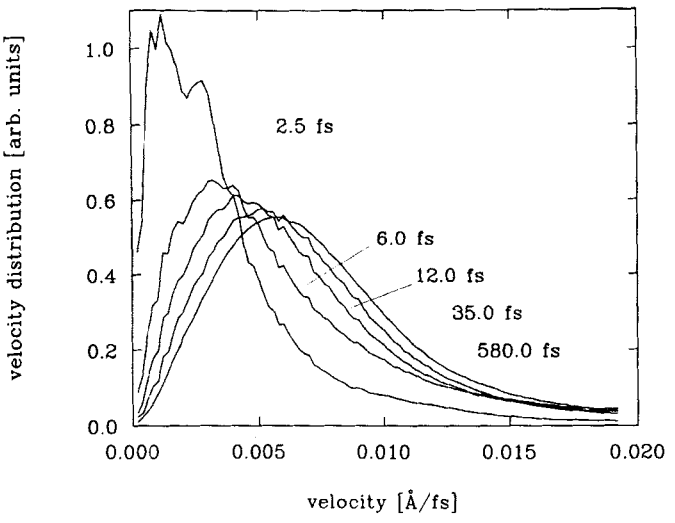

Figure 10 Velocity distribution of the simulation of the photosynthetic reaction center represented in Figure 9 at five different time instances: 2.5 fs, 6.0 fs, 12 fs, 35 fs, 580 fs.

Figure 9 presents the van der Waals energy, the Coulomb energy, the bond energy, and the kinetic energy of the photosynthetic reaction center as determined during a 1.5 ps simulation on our parallel computer. The results show that the van der Waals energy representing sterical interactions decreases rapidly, the Coulomb energy decreases slowly, and that the bond energy and kinetic energy increase. As explained above, the kinetic energy was kept from rising above a 300 K value by applying friction and by rescaling atomic velocities. The rescaling events are clearly discernable in Figure 9 in the short time regime. After 150 fs we did not scale velocities anymore and let the reaction center complex move freely. The resulting energies shown in Figure 9 are found, except for the Coulomb energy, to remain approximately constant, indicating that the complex reached equilibrium. The relaxation of the Coulomb energy is slower than the relaxation of the other degrees of freedom because of the relatively weak strength of the Coulomb forces and because the long range character requires a high degree of concertedness for protein motions to be effective in relaxing the Coulomb energy.

A more detailed test of equilibrium is furnished by the distribution of atomic velocities. We have monitored this distribution during the relaxation process. This distribution at representative instances in time, namely at 2.5 fs, 6.0 fs, 12 fs, 35 fs and 580 fs, is shown in Figure 10. Since we start with vanishing atomic velocities and, by rescaling, keep kinetic energy temperature constant at 300 K one expects the average velocities to increase monotonously to their 300 K value. This is born out, in fact, by the results shown in Figure 10.

The question arises to what extent the calculated asymptotic velocity distribution resembles a Maxwell distribution. A comparison between the velocity distribution of the reaction center atoms at 917 fs and a Maxwell distribution at 300 K for a single mass (13.2 amu, resulting from a numerical fit to the simulated velocity distribution), is shown in Figure 11. There is an obvious discrepancy between the two distributions.

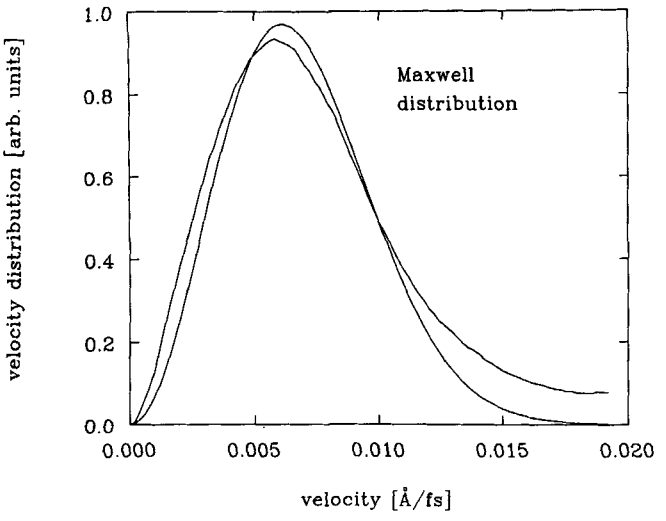

Figure 11 Comparison of the velocity distribution (at time $t = 917$ fs) resulting from a simulation of the photosynthetic reaction center and of a Maxwell distribution for a mass of 13.2 amu. This mass had been chosen to minimize the difference between the two distributions shown.

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

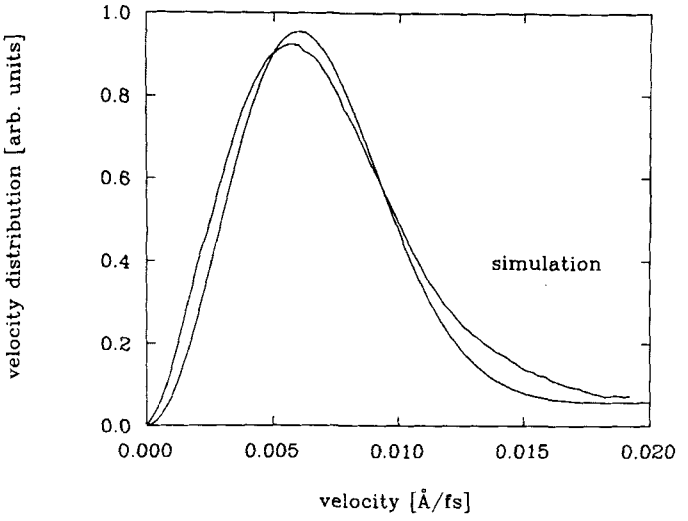

Figure 12 Comparison of the velocity distribution (at time $t = 917$ fs) resulting from a simulation of the photosynthetic reaction center and of the average Maxwell distribution evaluated according to Equation (18) for all reaction center atoms.

quinone, the strain being due to unfavorable interaction with some atoms of the protein. The strain was localized in a small section of the phytol chain, extending over six carbon atoms, the remaining part of the chromophore assumed a relaxed geometry. During the molecular dynamics simulation the secondary quinone was found to leave the photosynthetic reaction center rather rapidly, much faster than a linear extrapolation of Figure 13 would indicate. This finding seems to demonstrate that the

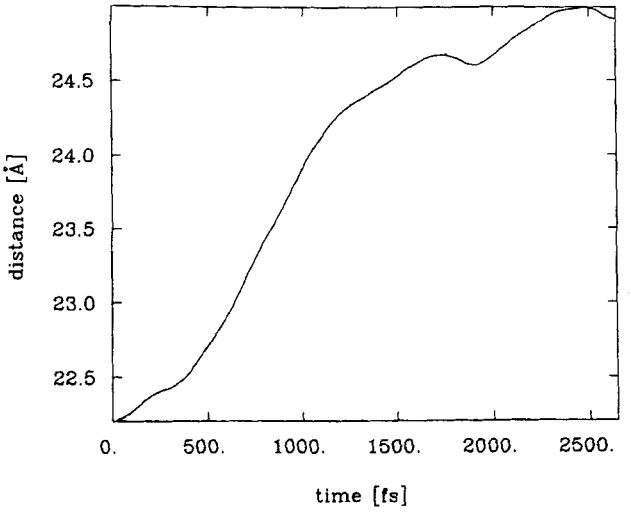

Figure 13 Time dependence of the distance between the center of mass of the reaction center and the center of the secondary quinone.

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

of Science and the Arts of the State of Bavaria, the Ministry for Science and Technology of the Federal Republic of Germany, as well as the German National Science Foundation (SFB 143, C1) for support in the initial phase of this work. Grants of computer time by the National Center for Supercomputing Applications and by the Max-Planck Society are gratefully acknowledged.

References

[1] H. Michel and J. Deisenhofer, "X-Ray diffraction studies on a crystalline bacterial photosynthetic reaction center: a progress report and conclusions on the structure of photosystem II reaction center", in *Encyclopedia of plant phsiology: photosynthesis III,* Vol. **19.** Springer, Berlin, 1986.

[2] H. Treutlein, "Molekulardynamiksimulation des Reaktionszentrums von Rhodopseudomonas viridis", PhD thesis, Technische Universität München, Physik-Department, T 30, James-Franck-Strasse, 8046 Garching, 1988.

[3] C. Niedermeier, "Elektrostatische Kontrolle der Primärprozesse im photosynthetischen Reaktionszentrum von Rhodopseudomonas viridis", Master's thesis, Technische Universität München, James-Franck-Strasse, 8046 Garching, 1989.

[4] A. Windemuth, "Dynamiksimulation von Makromolekülen", Master's thesis, Technische Universität München, Physik-Department, T 30, James-Franck-Strasse, 8046 Garching, August 1988.

[5] J. Deisenhofer and H. Michel, "The crystal structure of the photosynthetic reaction center from *Rhodopseudomonas viridis*", in *The photosynthetic bacterial reaction center: Structure and dynamics.* Plenum Press, London, 1987.

[6] B.R. Brooks, R.E. Bruccoleri, B.D. Olafson, D.J. States, S. Swaminathan, and M. Karplus, "CHARMM: a program for macromolecular energy, minimization, and dynamics calculations", *Journal of Computational Chemistry* **4**(2), 187 (1983).

[7] A.T. Brünger, "X-PLOR", The Howard Hughes Medical Institute and Department of Molecular Biophysics and Biochemistry, Yale University, 260 Whitney Avenue, P.O. Box 6666, New Haven, CT 06511, May 1988.

[8] A.T. Brünger, "Crystallographic refinement by simulated annealing", in *Crystallographic computing 4: Techniques and new technologies.* Clarendon Press, Oxford, 1988.

[9] "OCCAM2 reference manual", Prentice Hall, Cambridge University Press, Cambridge, 1988, #720CC4501.

[10] D. May, "OCCAM2, Language Definition". INMOS, February 1987, #720CC04401, #720CC04402.

[11] D. Pountain, "A Tutorial Introduction to OCCAM Programming". INMOS, April 1987, #72TDS11500.

[12] R. Steinmetz, "OCCAM 2". Hüthig-Verlag, Heidelberg, 1987.

[13] D.C. Rapaport, "Large-scale molecular dynamics simulation using vector and parallel computers", *Computer Physics Reports,* **9,1** (1988).

[14] D. Fincham, "Parallel computers and molecular simulation", *Molecular Simulation,* **1**, 1 (1987).

[15] M.K. Ch.L. Brooks III and B.M. Pettitt, "Proteins: A Theoretical Perspective of Dynamics, Structure and Thermodynamics". John Wiley, New York, 1988.

[16] L. Verlet, "Computer 'experiments' on classical fluids. I. Thermodynamical properties of Lennard-Jones molecules", *Physical Review,* **159,** 98 (1967).

[17] L. Greengard and V. Rokhlin, "On the efficient implementation of the fast multipole algorithm", Technical Report Research Report 602, Yale Department of Computer Science, 1988, and references therein.

[18] I. Klapper, R. Hagstrom, R. Fine, K. Sharp, and B. Honig, "Focusing of electric fields in the active site of Cu-Zn superoxide dismutase: Effects of ionic strength and amino-acid modification", *Proteins: Structure, Function, and Genetics,* **1**, 47-59, (1986).

[19] N.S. Ostlund and R.A. Whiteside, "A machine architecture for molecular dynamics: The systolic loop", in: B. Venkataraghavan and R.J. Feldmann, eds, "Macromolecular Structure and Specificity: Computer-assisted Modeling and Applications", pages 192-208. The New York Academy of Sciences, New York, New York, 1985, Annals of the New York Academy of Sciences.

[20] W.D. Hillis and Barnes, "Programming a highly parallel computer", *Nature,* **326,** 27 (1987).

[21] "INMOS Spectrum", INMOS GmbH, Danziger Str. 2, 8057 Eching, 1988.

[22] INMOS, "Transputer Development System", 1988, #72TRN01100.

[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.