JCTC Journal of Chemical Theory and Computation

Implementation of Molecular Dynamics and Its Extensions with the Coarse-Grained UNRES Force Field on Massively Parallel Systems: Toward Millisecond-Scale Simulations of Protein Structure, Dynamics, and Thermodynamics

Adam Liwo,*[,†,‡] Stanisław Ołdziej,[‡,§] Cezary Czaplewski,[†,‡] Dana S. Kleinerman,[‡] Philip Blood,[||] and Harold A. Scheraga[‡]

Faculty of Chemistry, University of Gdańsk, Sobieskiego 18, 80-952 Gdańsk, Poland, Baker Laboratory of Chemistry and Chemical Biology, Cornell University, Ithaca, New York 14853-1301, Laboratory of Biopolymer Structure, Intercollegiate Faculty of Biotechology, University of Gdańsk, Medical University of Gdańsk, Kładki 24, 80-822 Gdańsk, Poland, and Pittsburgh Supercomputing Center, Carnegie Mellon University, University of Pittsburgh, 300 S. Craig Street, Pittsburgh, Pennsylvania 15213

Received August 4, 2009

Abstract: We report the implementation of our united-residue UNRES force field for simulations of protein structure and dynamics with massively parallel architectures. In addition to coarse-grained parallelism already implemented in our previous work, in which each conformation was treated by a different task, we introduce a fine-grained level in which energy and gradient evaluation are split between several tasks. The Message Passing Interface (MPI) libraries have been utilized to construct the parallel code. The parallel performance of the code has been tested on a professional Beowulf cluster (Xeon Quad Core), a Cray XT3 supercomputer, and two IBM BlueGene/P supercomputers with canonical and replica-exchange molecular dynamics. With IBM BlueGene/P, about 50% efficiency and a 120-fold speed-up of the fine-grained part was achieved for a single trajectory of a 767-residue protein with use of 256 processors/trajectory. Because of averaging over the fast degrees of freedom, UNRES provides an effective 1000-fold speed-up compared to the experimental time scale and, therefore, enables us to effectively carry out millisecond-scale simulations of proteins with 500 and more amino acid residues in days of wall-clock time.

1. Introduction

Simulations of conformational changes in proteins and dynamics of protein conformations are nowadays of great importance in biochemistry, biophysics, and medical sciences.[1−13] All-atom molecular dynamics (MD) *ab initio* folding simulations (which require at least a microsecond time scale) are still restricted to the nanosecond time scale for large proteins and are possible only for proteins with lengths up to 60 residues with implicit-solvent approaches (however, simulations of the dynamics of big proteins starting from the experimental structure have been carried out since the early 1990s[6]), although great progress has been made with distributed computing (the FOLDING@HOME project);[14] the creation of very efficient load-balanced parallel codes such as GROMACS,[15] NAMD,[16] or DESMOND;[17] and, very recently, with the implementation of all-atom MD programs

* Corresponding author phone: +48 58 523 5430, fax: +48 58 523 5472, e-mail: adam@chem.univ.gda.pl.

[†] Faculty of Chemistry, University of Gdańsk.

[‡] Cornell University.

[§] Medical University of Gdańsk.

[||] Pittsburgh Supercomputing Center.

on graphical processor units (GPUs)[18] and the construction of dedicated machines.[19] For recent developments of the software for all-atom MD simulations, see the latest review by Klepeis et al.[20] Coarse-grained models have, therefore, become of great importance in the field.[21−26] In the past decade, we have been developing a physics-based model of polypeptide chains, hereafter referred to as UNRES (for UNited RESidue).[27−39]

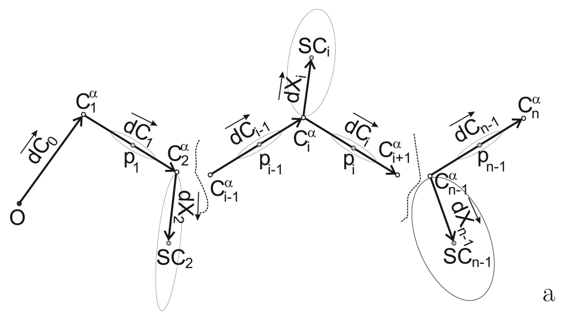
UNRES has been applied with considerable success in energy-based protein-structure prediction[40] and was later extended[41−43] to simulations of protein-folding pathways by implementing a coarse-grained dynamics approach, providing a 4000-fold speed-up compared to all-atom MD with an explicit solvent.[42,43] This speed-up factor was obtained by comparing the mean first passage time (MFPT) of the simulated folding of deca-alanine obtained with UNRES to that obtained with AMBER with explicit water. However, this speedup is not as big for larger proteins for which the water layer constitutes a smaller fraction of the system; also, introducing a cutoff on nonbonded interactions in all-atom simulations (which has not yet been performed for UNRES) causes a major reduction of the computation time. With the UNRES/MD approach, *ab initio* folding simulations of 75-residue proteins are possible within hours of single-processor time[43] (compared to weeks for all-atom MD with implicit solvent). Effectively, proteins fold in nanoseconds[42,43] with UNRES, while the shortest experimental protein-folding times are on the order of microseconds.[44] Consequently, UNRES provides a ~1000-fold speed-up with respect to the experimental time scale. It should be noted that the coarse graining of protein representation, in particular using implicit solvent, reduces the accuracy of simulations; however, they still enable us to draw meaningful conclusions regarding protein folding. Trivial parallelization of molecular dynamics simulations, in which a single trajectory is calculated independently by a given processor, has enabled us[45] to study the folding kinetics of protein A, without biasing the potential toward the native structure, as in similar other studies.[46,47] With the aid of replica-exchange (REMD)[48] and multiplexed replica exchange molecular dynamics (MREMD),[49] we are now able to simulate protein-folding thermodynamics and perform ensemble-based prediction of protein structure.[36,37] We parallelized these algorithms,[50,51] and they scale well even for thousands of trajectories. Because of infrequent communication characteristic of the REMD and MREMD algorithms, they perform equally well on Beowulf clusters and on supercomputers with a fast interconnect. However, even with this reduction of computational time, *ab initio* folding simulations of proteins with a size of about 150 amino acid residues take weeks, and those of larger proteins are still out of reach in reasonable time, if a single processor handles the energy and gradient evaluation of a given conformation.

In this paper, we report the extension of UNRES to large protein systems, by introducing fine-grained parallelization of energy and gradient evaluation and other elements of an elementary step of molecular dynamics, in addition to *coarse-grained* parallelization, which controls it; this means that a group of fine-grained tasks is included in a coarse-grained task, which is dedicated to a given conformation. We demonstrate that, by setting 50% efficiency as a reasonable cutoff for the performance of a fine-grained-parallelized algorithm, it is reasonable to run calculations involving fine-grained parallelization with 8−16 processors dedicated to a single conformation for proteins with a size of more than 60 amino acid residues on professional Beowulf clusters and on the Cray XT3, which have faster processors (and thus less computation time per processor), and, for proteins with 30 amino acid residues or more, calculations on the IBM BlueGene/P, which has slower processors (and thus more computation time per processor) together with a fast interconnect, are reasonably efficient. For proteins with a size of about 800 amino acid residues, the achievable speedup is about 20-fold with professional Beowulf clusters, 30-fold with the Cray XT3, and over 100-fold with IBM BlueGene/P. This enables us to run *ab initio* simulations of folding dynamics and thermodynamics of such proteins in days of wall-clock time provided that massively parallel resources are available.

This paper is organized as follows. In sections 2.1, 2.2, and 2.3, we give a brief summary of the UNRES force field and MD and its extensions with UNRES. In section 2.4, we characterize the programming environment and machines used in this study. In section 2.5, we describe the parallelization scheme and its implementation. In section 3, we report the performance of the fine-grained code, and finally, in section 4, we discuss the implications of our work in biomolecular simulations and possible extensions of the approach, including implementation of the code on the GPUs.

2. Methods

2.1. The UNRES Force Field. In the UNRES model,[27−39] a polypeptide chain is represented by a sequence of α-carbon (C^α) atoms linked by virtual bonds with attached united side chains (SC) and united peptide groups (p). Each united peptide group is located in the middle between two consecutive α-carbons. Only these united peptide groups and the united side chains serve as interaction sites, the α-carbons serving only to define the chain geometry, as shown in Figure 1. The UNRES force field has been derived as a restricted free energy (RFE) function[29,30] of an all-atom polypeptide chain plus the surrounding solvent, where the all-atom energy function is averaged over the degrees of freedom that are lost when passing from the all-atom to the simplified system (viz., the degrees of freedom of the solvent, the dihedral angles χ for rotation about the bonds in the side chains, and the torsional angles λ for rotation of the peptide groups about the $C^\alpha \cdots C^\alpha$ virtual bonds).[52] The RFE is further decomposed into factors coming from interactions within and between a given number of united interaction sites.[30] Expansion of the factors into generalized Kubo cumulants[53] enabled us to derive approximate analytical expressions for the



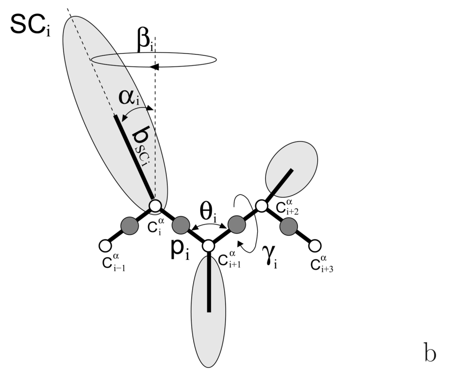


Figure 1. The UNRES model of the polypeptide chains. (a) Illustration of the variables used in MD. The terminal residues and a residue inside the chain are shown. The C^α atoms, the centers of the peptide groups (p), and side-chain centers (SC) are indicated by small open circles, while open ellipses centered at the p and SC centers indicate that these sites interact through noncentral forces. The C^α atoms serve only as geometric points and are not interaction sites. The peptide groups are positioned halfway between two consecutive C^α's. The geometry of the chain is defined by the vector $\vec{\mathbf{dC}}_0$, defining the position of the first C^α atom (pointing from the origin O of the reference system to this atom), by backbone virtual bonds ($\vec{\mathbf{dC}}_1$, $\vec{\mathbf{dC}}_2$, ..., $\vec{\mathbf{dC}}_{n-1}$), and side-chain ($\vec{\mathbf{dX}}_2$, $\vec{\mathbf{dX}}_3$, ..., $\vec{\mathbf{dX}}_{n-1}$) vectors. For Gly residues, the respective SC atoms are methylene groups and are located exactly at the corresponding C^α atoms; therefore, there are no $\vec{\mathbf{dX}}$ vectors for these residues. The C^α_1 or C^α_n atoms are either the terminal methyl groups (in these cases, they are treated as Gly "side chains") if a chain is terminally blocked or dummy atoms if no blocking groups are present. (b) Illustration of internal coordinates pertaining to the *i*th residue used in eq 1: backbone virtual-bond-valence angles (θ_i), backbone virtual-bond-dihedral angle (γ_i), side-chain virtual-bond length (b_{SC_i}), and the angles α_{SC_i} and β_{SC_i} defining the position of the *i*th side chain with respect to the local coordinate frame defined by C^α_{i-1}, C^α_i, and C^α_{i+1}. All peptide groups are assumed to be in a trans configuration with an equilibrium virtual-bond length of 3.8 Å.

respective terms,[29,30] including the *multibody* or *correlation* terms, which are derived in other force fields from structural databases or on a heuristic basis.[54] The theoretical basis of the force field is described in detail in our earlier paper.[30]

The energy of the virtual-bond chain is expressed by eq 1.

$$
\begin{aligned}
U = {} & w_{\mathrm{SC}} \sum_{i<j} U_{\mathrm{SC}_i\mathrm{SC}_j} + w_{\mathrm{SCp}} \sum_{i\neq j} U_{\mathrm{SC}_i\mathrm{p}_j} + w_{\mathrm{pp}}^{\mathrm{VDW}} \sum_{i<j-1} U_{\mathrm{p}_i\mathrm{p}_j}^{\mathrm{VDW}} + w_{\mathrm{pp}}^{\mathrm{el}} f_2(T) \sum_{i<j-1} U_{\mathrm{p}_i\mathrm{p}_j}^{\mathrm{el}} + \\
& w_{\mathrm{tor}} f_2(T) \sum_i U_{\mathrm{tor}}(\gamma_i) + w_{\mathrm{tord}} f_3(T) \sum_i U_{\mathrm{tord}}(\gamma_i, \gamma_{i+1}) + w_{\mathrm{b}} \sum_i U_{\mathrm{b}}(\theta_i) + w_{\mathrm{rot}} \sum_i U_{\mathrm{rot}}(\alpha_{\mathrm{SC}_i}, \beta_{\mathrm{SC}_i}) + \\
& w_{\mathrm{bond}} \sum_i U_{\mathrm{bond}}(d_i) + w_{\mathrm{corr}}^{(3)} f_3(T) U_{\mathrm{corr}}^{(3)} + w_{\mathrm{corr}}^{(4)} f_4(T) U_{\mathrm{corr}}^{(4)} + \\
& w_{\mathrm{corr}}^{(5)} f_5(T) U_{\mathrm{corr}}^{(5)} + w_{\mathrm{corr}}^{(6)} f_6(T) U_{\mathrm{corr}}^{(6)} + w_{\mathrm{turn}}^{(3)} f_3(T) U_{\mathrm{turn}}^{(3)} + \\
& w_{\mathrm{turn}}^{(4)} f_4(T) U_{\mathrm{turn}}^{(4)} + w_{\mathrm{turn}}^{(6)} f_6(T) U_{\mathrm{turn}}^{(6)}
\end{aligned}
\qquad (1)
$$

where θ_i is the backbone virtual-bond angle, γ_i is the backbone virtual-bond-dihedral angle, α_i and β_i are the angles defining the location of the united side-chain center of residue *i* (Figure 1b), and d_i is the length of the *i*th virtual bond, which is either a $C^\alpha\cdots C^\alpha$ virtual bond or a $C^\alpha\cdots$SC virtual bond. Each term is multiplied by an appropriate weight, w_x, and the terms corresponding to factors of an order higher than 1 are additionally multiplied by the respective temperature factors, which were introduced in our recent work[36] and which reflect the dependence of the first generalized-cumulant term in those factors on temperature, as discussed in refs 36 and 55. The factors f_n are defined by eq 2.

$$
f_n(T) = \frac{\ln[\exp(1) + \exp(-1)]}{\ln\{\exp[(T/T_o)^{n-1}] + \exp[-(T/T_o)^{n-1}]\}} \qquad (2)
$$

where $T_o = 300$ K.

The term $U_{\mathrm{SC}_i\mathrm{SC}_j}$ represents the mean free energy of the hydrophobic (hydrophilic) interactions between the side chains, which implicitly contain the contributions from the interactions of the side chain with the solvent. The term $U_{\mathrm{SC}_i\mathrm{p}_j}$ denotes the excluded-volume potential of the side-chain−peptide-group interactions. The peptide-group interaction potential is split into two parts: the Lennard-Jones interaction energy between peptide-group centers ($U_{\mathrm{p}_i\mathrm{p}_j}^{\mathrm{VDW}}$) and the average electrostatic energy between peptide-group dipoles ($U_{\mathrm{p}_i\mathrm{p}_j}^{\mathrm{el}}$); the second of these terms accounts for the tendency to form backbone hydrogen bonds between peptide groups p_i and p_j. The terms U_{tor}, U_{tord}, U_{b}, U_{rot}, and U_{bond} are the virtual-bond-dihedral angle torsional terms, virtual-bond dihedral angle double-torsional terms, virtual-bond angle bending terms, side-chain rotamer, and virtual-bond-deformation terms; these terms account for the local propensities of the polypeptide chain. The terms $U_{\mathrm{corr}}^{(m)}$ represent correlation or multibody contributions from the coupling between backbone-local and backbone-electrostatic interactions, and the terms $U_{\mathrm{turn}}^{(m)}$ are correlation contributions involving *m* consecutive peptide groups; they are, therefore, termed turn contributions. In the present version of UNRES, the terms of an order higher than 4 are not present because, in our earlier work,[34] we found that the correlation terms of an order up to 4 are sufficient to reproduce regular secondary structures, while including higher-order terms results in a substantial increase of the cost of energy and force evaluation. The multibody terms are indispensable for reproduction of regular α-helical and β-sheet structures.[29,30,54]

The internal parameters of $U^{\mathrm{VDW}}_{p_ip_j}$, $U^{\mathrm{el}}_{p_ip_j}$, U_{tor}, U_{tord}, U_{b}, $U^{(m)}_{\mathrm{corr}}$, and $U^{(m)}_{\mathrm{turn}}$ were derived by fitting the analytical expressions to the RFE surfaces of model systems computed at the MP2/6-31G** *ab initio* level;[32,33,35] the parameters of U_{bond} and U_{rot} were derived similarly from the energy surfaces of terminally blocked amino acid residues calculated with the AM1 semiempirical method,[38,39] while the parameters of $U_{\mathrm{SC}_i\mathrm{SC}_j}$ and $U_{\mathrm{SC}_ip_j}$ were derived by fitting the calculated distribution functions to those determined from the PDB.[28] The *w*'s are the weights of the energy terms, and they can be determined only by optimization of the potential-energy function, as described in our earlier work.[31,34,36] In the present study, we use a recent version of UNRES developed by extensive random search of the energy-term weights.[56]

2.2. Molecular Dynamics with UNRES. In this section, we summarize briefly the theory behind the UNRES/MD approach and recall the equations necessary to understand its algorithmic aspects. The reader is referred to our earlier work[41,42] for details. The equations of motion for the UNRES chain are Langevin-dynamics equations because the solvent is implicit in UNRES. Consequently, it contributes to conservative forces (through the RFE) and gives rise to nonconservative forces which originate in energy exchange of the polypeptide chain with the solvent (the stochastic and friction forces). Because the geometry of an UNRES chain is not uniquely defined by the Cartesian coordinates of the interacting sites, we chose the virtual-bond vectors $C^\alpha_i \cdots C^\alpha_{i+1}$ (denoted as $\mathbf{dC}_i$) and $C^\alpha_i \cdots SC_i$ (denoted as $\mathbf{dX}_i$) as generalized coordinates, as expressed by eq 3.

$$\mathbf{q} = (\mathbf{dC}_0, \mathbf{dC}_1, ..., \mathbf{dC}_{n-1}, \mathbf{dX}_s, \mathbf{dX}_{s+1}, ..., \mathbf{dX}_e)^T \quad (3)$$

where *s* and *e* are the indices of the first and of the last side-chain vectors, respectively (i.e., of the first and of the last nonglycine residues; the glycine residues carry no **dX** vectors). The vector $\mathbf{dC_0}$ specifies the position of the first C^α atom of the chain. The vectors $\dot{\mathbf{q}}$ and $\ddot{\mathbf{q}}$ denote generalized velocities and generalized accelerations, respectively We assume that the virtual bonds are elastic rods with mass distribution that scales with the length of a rod.[41] The Cartesian coordinates of the interacting sites, p_i and SC_i, defined by eq 4

$$\mathbf{x} = (\mathbf{r}_{p_1}, \mathbf{r}_{p_1}, ..., \mathbf{r}_{p_{n-1}}, \mathbf{r}_{SC_1}, \mathbf{r}_{SC_2}, ..., \mathbf{r}_{SC_n})^T \quad (4)$$

are related to the generalized coordinates by a linear transformation expressed by eq 5

$$\mathbf{x} = \mathbf{A}\mathbf{q} \quad (5)$$

where **A** is a constant matrix such that $a_{i(k),j} = 0$ [$i(k)$ being a Cartesian coordinate of site *k*] if the coordinates up to *j* correspond to virtual-bond vectors of the part of the chain to the right of site *k*, $a_{i(k),j} = 1$ if the coordinate corresponds to virtual-bond vector to the left of site *k* or to a $C^\alpha \cdots SC$ virtual bond containing the side chain with index *k*, and $a_{i(k),j} = 1/2$ if the coordinate corresponds to the virtual-bond vector containing the peptide group with index *i*, as expressed by eq 6.[41] The same relationship holds between the time derivatives of **x** and **q**.

$$\mathbf{A} = \begin{pmatrix} \mathbf{1} & \frac{1}{2}\mathbf{1} \\ \mathbf{1} & \mathbf{1} & \frac{1}{2}\mathbf{1} \\ \vdots & \vdots & \vdots & \ddots \\ \mathbf{1} & \mathbf{1} & \mathbf{1} & \dots & \frac{1}{2}\mathbf{1} \\ \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & h_{SC_s}\mathbf{1} \\ \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & h_{SC_{s+1}}\mathbf{1} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots \\ \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \dots & \mathbf{1} & \mathbf{0} & \mathbf{0} & \dots & h_{SC_e}\mathbf{1} \end{pmatrix} \quad (6)$$

where h_{SC_i} is 0 if SC_i = Gly and is 1 otherwise, **1** is a 3 × 3 identity matrix, and **0** is a 3 × 3 matrix composed of zeros. It should be noted that the columns of matrix **A** run through all variables [the elements of the vector **q** of eq 3], and the rows run through the Cartesian coordinates of the interacting sites.

The peptide groups and side chains are represented as stretchable rods with uniformly distributed masses.[41] The Langevin equation for UNRES[42] is given by eq 7

$$\mathbf{F} = \mathbf{G}\ddot{\mathbf{q}} = -\nabla_{\mathbf{q}}U(\mathbf{q}) - \mathbf{A}^{\mathbf{T}}\Gamma\mathbf{A}\dot{\mathbf{q}} + \mathbf{A}^{\mathbf{T}}\mathbf{f}^{\mathbf{rand}} \quad (7)$$

$$\mathbf{G} = \mathbf{A}^{\mathbf{T}}\mathbf{M}\mathbf{A} + \mathbf{H} \quad (8)$$

where **M** is the diagonal matrix of the masses of the sites (united peptide groups and united side chains) defined by eq 9, **H** [a diagonal matrix defined by eq 10] is the part of the inertia matrix corresponding to the internal stretching motion of the virtual bonds,[41] **G** [defined by eq 8] is the inertia matrix, Γ is the diagonal friction tensor (represented by the friction matrix with elements γ_{ii} determined by Stokes law) acting on the interacting sites such that γ_{ii} is the coefficient of the site corresponding to the *i*th coordinate, $\mathbf{f}^{\mathbf{rand}}$ is the vector of random forces acting on interacting sites, *U* is the UNRES potential energy defined by eq 1, $\nabla_{\mathbf{q}}$ denotes the gradient with respect to **q**, and **F** is the vector of forces. In this work, we did not use the stochastic and friction terms; therefore constant temperature was maintained by using the Berendsen thermostat[57] implemented in UNRES/MD in our earlier work.[42]

$$\mathbf{M} = \begin{pmatrix} m_p\mathbf{1} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & m_p\mathbf{1} & \dots & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & m_{SC_s}\mathbf{1} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & m_{SC_{s+1}}\mathbf{1} & \dots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \dots & m_{SC_e}\mathbf{1} \end{pmatrix} \quad (9)$$

where m_p is the mass of a peptide group and m_{SC_i} is the mass of the *i*th side chain.

The matrix **H** is defined by eq 10.

$$\mathbf{H} = \begin{pmatrix} I_p\mathbf{1} & \mathbf{0} & \ldots & \mathbf{0} & \mathbf{0} & \ldots & \mathbf{0} \\ \mathbf{0} & I_p\mathbf{1} & \ldots & \mathbf{0} & \mathbf{0} & \ldots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \ldots & I_{SC_{i_1}}\mathbf{1} & \mathbf{0} & \ldots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \ldots & \mathbf{0} & I_{SC_{i_2}}\mathbf{1} & \ldots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \ldots & \mathbf{0} & \mathbf{0} & \ldots & I_{SC_{i_m}}\mathbf{1} \end{pmatrix} \quad (10)$$

with[42]

$$I_p = (1/12)m_p \text{ and } I_{SC_i} = (1/3)m_{SC_i} \quad (11)$$

To compute the accelerations from forces by using eq 7 requires matrix multiplication of the right-hand side of this equation by $\mathbf{G}^{-1}$ [eq 12].

$$\ddot{\mathbf{q}} = \mathbf{G}^{-1}\mathbf{F} \quad (12)$$

where **F** is the vector of forces defined by the right-hand side of eq 7. Because **G** is a constant matrix, its inverse [which involves $\mathcal{O}(n^3)$ computational effort] can be computed only once at the beginning of a simulation and used later. Moreover, it can be seen from eqs 6, 9, and 10 that, if the coordinates in vectors **x** and **q** are rearranged to group all x, then all y, and finally, all z coordinates and the rows and columns of the matrix **G** are rearranged accordingly, matrix **G** is separated into three blocks corresponding to x, y, and z parts, thereby reducing the number of operations in matrix multiplication by a factor of 3.

2.3. Replica-Exchange and Multiplexing-Replica-Exchange Molecular Dynamics. To sample the conformational space more efficiently than by canonical MD, we extended[50,51] the UNRES/MD approach with the replica-exchange[48] and multiplexing replica-exchange[49] molecular dynamics method. In this method, M canonical MD simulations are carried out simultaneously, each one at a different temperature. Initially the temperatures increase with the sequential number of simulations (trajectories). After every $m < M$ step, an exchange of temperatures between neighboring trajectories (in the order from 1 to M) is attempted, the decision about the exchange being made based on the Metropolis criterion which, taking into account the temperature dependence of the force field, is expressed by eq 13.

$$\Delta = [\beta_{i+1}U(\mathbf{X}_{i+1},\beta_{i+1}) - \beta_iU(\mathbf{X}_{i+1},\beta_i)] - [\beta_{i+1}U(\mathbf{X}_i,\beta_{i+1}) - \beta_iU(\mathbf{X}_i,\beta_i)],\ i = 1, 2, \ldots, M \quad (13)$$

where $\beta_i = 1/RT_i$, T_i being the absolute temperature corresponding to the ith trajectory, and $\mathbf{X}_i$ denotes the variables of the UNRES conformation of the ith trajectory at the attempted exchange point. If $\Delta \le 0$, T_i and T_{i+1} are exchanged; otherwise the exchange is performed with probability $\exp(-\Delta)$.

The multiplexing variant of the REMD method (MREMD)[49] differs from the REMD method in that several trajectories are run at a given temperature. Each set of trajectories run at a different temperature constitutes a *layer*. Exchanges are attempted not only within a single layer but also between layers. In our very recent study,[51] we demonstrated that such a procedure increases the power of REMD very considerably, and convergence of the thermodynamic quantities is achieved much faster.

2.4. Programming Languages, Libraries, Other Software Used, and Machines. As in our previous work on the implementations of the conformational space annealing[58,59] and the REMD[50] and MREMD[51] versions of UNRES, the Message Passing Interface (MPI) library was implemented to construct the parallel code. Therefore, the description of parallelization in section 2.5 is message-passing oriented. The code was written in our laboratory, using the FORTRAN 77 programming language, and was parallelized using the MPI library. Standard blocking MPI_SCATTER and MPI_GATHER MPI library routines were implemented to send the data from a master task to the slave task and to collect data from the slave tasks, respectively, and MPI_REDUCE with the MPI_SUM operator was used to sum the contributions to energy, energy gradients, and results of matrix-vector multiplications from slave tasks at a fine-grain master task. The coarse-grained tasks and each of the fine-grained tasks formed separate MPI communicators to avoid possible message interception.[60]

For storage of trajectory files, we use the freely available Europort Data Compression XDRF library (http://hpcv100.rc.rug.nl/xdrfman.html). The BLAS routines[61] are used for matrix diagonalization.

The UNRES code with MD and MREMD has been implemented and tested on the Xeon Quad Core professional Beowulf cluster at the Academic Computer Center in Gdańsk, TASK (galera.task.gda.pl; 1344 Intel Xeon Quad-Core processors, 5376 cores, Mellanox InfiniBand interconnect with 20 Gb/s bandwidth; Web page at http://www.task.gda.pl/english/kdm.html), on the Cray XT3 supercomputer at the Pittsburgh Supercomputer Center [bigben.psc.edu; 4096 processors (dual-core), the 3D torus network; Web page at http://www.psc.edu/general/hardware.php], and on the following two IBM BlueGene/P massively parallel machines: at Argonne National Laboratory (intrepid.alcf.anl.gov; 40 960 quad-core compute nodes; Web page at http://www.alcf.anl.gov/resources/storage.php) and at the Jülich Supercomputer Center (jugene.fz-juelich.de; 73 728 compute nodes, each with a four-way 32-bit PowerPC 450 core 850 MHz processor, three-dimensional torus network; Web page at http://www.fz-juelich.de/jsc/jugene). Of these four machines, galera.task.gda.pl has the largest single-processor speed, while both IBM BlueGene/P machines have the fastest and the most efficient communication but the slowest processors.

2.5. Parallelization and Parallel Programming System Used. *2.5.1. General Parallelization Scheme.* As mentioned in the Introduction, the code is parallelized at a two-grain level:

(1) Coarse-grained level, in which a given coarse-grained task (CG) is assigned to a given trajectory. In a multitrajectory canonical molecular dynamics run, there is no communication between tasks except synchronization at the end

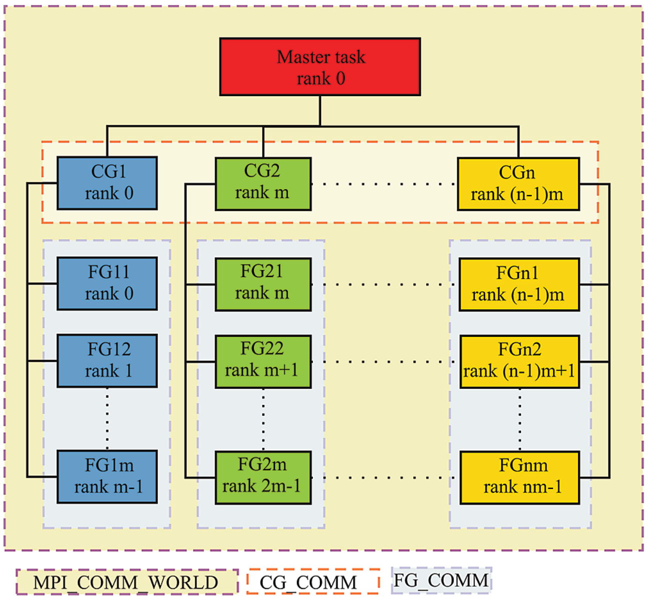


Figure 2. Scheme for the distribution of *n* coarse-grained (CG1, CG2, ..., CGn) and *m* fine-grained tasks for each coarse-grained task (FG11, FG12, ...FG1m, FG 21..., FGnm) among *nm* processes (with ranks 0, ..., nm − 1) and of communicators (shown as boxes bordered by dashed lines) pertaining to the respective tasks. MPI_COMM_WORLD is the communicator containing all processes. CG_COMM contains the processes assigned to coarse-grained tasks, and FG_COMM is the communicator containing processes assigned to a given fine-grained task.

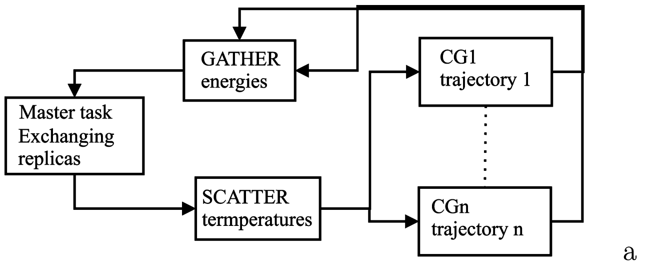
of the job. For REMD, communication occurs at the time of exchanging replicas (see section 2.3).

(2) Fine-grained level, in which (i) computation of energy and forces, and (ii) computation of accelerations from forces [eq 7] are distributed to fine-grained tasks (FGs).

A scheme of tasks and processor assignment to tasks by rank is shown in Figure 2. Figure 3a and b shows the basic operations and communication between the master task and CG tasks in a replica-exchange run, and between a CG task and the pertinent FG tasks in energy and gradient evaluation.

The coarse-grained parallelism is the same as discussed in our earlier work;[50,51] therefore, we provide only a brief summary here.

As discussed in our earlier work,[51] one problem with the efficiency of parallel (M)REMD runs involving over 500 trajectories run simultaneously is the rapid loss of scalability because of synchronization implied in classical (M)REMD algorithms at each exchange. Even if the parallel system is homogeneous, 100% of processor time is never available for a compute task, which results in a slightly different processor speed, depending on the task. Moreover, a distance cutoff is applied in the computation of four-body correlation interactions,[29] and in our A-MTS algorithm[62] for integrating the equations of motion, the split number of a given time step depends on current conformation. Therefore, the computational effort depends on the trajectory. The discrepancy between the slowest and the fastest trajectories increases with increasing number of trajectories. Therefore, we modified[51] the (M)REMD algorithm to reduce synchronization overhead. As soon as a processor has executed the preset number of



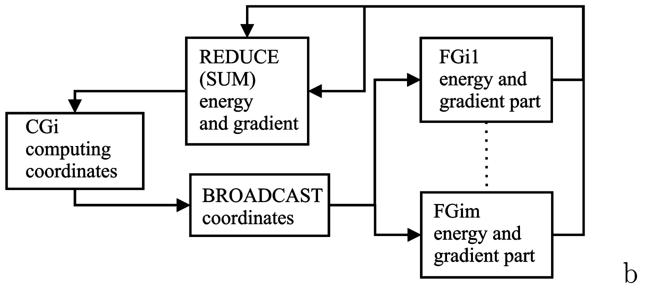

REDUCE (SUM) energy and gradient
CGi computing coordinates
BROADCAST coordinates
FGi1 energy and gradient part
FGim energy and gradient part
b

Figure 3. Basic operations and communication between the master task and the CG tasks (a), and between a CG task and the pertinent FG tasks (b).

steps since the last exchange, it sends a message to the other processors to perform replica exchange. With this modification, we reached over 70% scalability for 4096 trajectories.[51]

The fine-grained parallelism introduced in this work is discussed in detail in the next subsections.

2.5.2. Complexity of Operations Involved in a Single MD Step. In order to decide about the parallelization strategy, the complexity of the operations involved in a single MD step must be analyzed. In all-atom MD simulations, almost all of the computational effort is connected with energy and force evaluation; however, in UNRES, the calculation of acceleration from forces involves a matrix−vector multiplication [eq 12], gradient transformation (discussed in section 2.5.4), and other operations specific for coarse-grained approaches.

The complexity and relative contributions to single-processor wall-clock time [calculated for the 1TF5 protein (767 residues, $\alpha + \beta$)] of the energy terms and other operations performed during the production phase of an UNRES/MD run are summarized in Table 1. Because of the significant reduction of the number of interaction sites compared to an all-atom representation of polypeptide chains, we did not yet introduce a cutoff on nonbonded interactions, as opposed to all-atom MD software[15−17] (although we are planning to introduce the cutoff in the future). Therefore, the complexity of all long-range interactions is $\mathcal{O}(n^2)$, n being the number of residues (Table 1) and not $\mathcal{O}(n)$. It can be seen from Table 1 that the greatest effort is required to compute the average-electrostatic ($U^{\mathrm{el}}_{p_ip_j}$) and van der Waals ($U^{\mathrm{VDW}}_{p_ip_j}$) as well as third-order multibody ($U^{(3)}_{\mathrm{corr}}$) interactions between the peptide groups; individual timings cannot be provided because these terms are computed in a single subroutine and use common intermediate quantities (such as, e.g., the distance between peptide-group centers). The next are the $U_{\mathrm{SC}_i\mathrm{SC}_j}$ and $U_{\mathrm{SC}_ip_j}$ terms, and the least expensive

Table 1. Complexity and Timing of Different Operations of UNRES for the 1TF5 Protein Obtained with 1, 32, and 128 Processors at bigben.psc.edu[a]

operation	complexity	parallelized	% contribution 1 CPU	32 CPUs	128 CPUs
U_{bond}	$\mathcal{O}(n)$	yes	0.0	0.0	0.0
U_b	$\mathcal{O}(n)$	yes	0.4	0.2	0.1
U_{rot}	$\mathcal{O}(n)$	yes	0.2	0.1	0.1
U_{tor}	$\mathcal{O}(n)$	yes	0.1	0.0	0.0
U_{tord}	$\mathcal{O}(n)$	yes	0.5	0.0	0.1
$U_{turn}^{(3)}$	$\mathcal{O}(n)$	yes	0.0	0.0	0.0
$U_{turn}^{(4)}$	$\mathcal{O}(n)$	yes	0.1	0.1	0.1
$U_{SC_iSC_j}$	$\mathcal{O}(n^2)$	yes	21.2	9.5	5.4
$U_{SC_ip_j}$	$\mathcal{O}(n^2)$	yes	8.0	3.6	2.1
$U_{p_ip_j}^{el+VDW} + U_{corr}^{(3)}$	$\mathcal{O}(n^2)$	yes	62.6	29.3	18.6
$U_{corr}^{(4)b}$	$\mathcal{O}(n^2)$	yes	3.5	6.3	8.5
auxiliary to compute $U_{corr}^{(3)}$ and $U_{turn}^{(3,4)}$	$\mathcal{O}(n)$	attempted[d]	0.1	3.0	5.0
summing energy components[b]	$\mathcal{O}(n)$	yes	0.0	18.0	11.5
gradient transform[b]	$\mathcal{O}(n^2)$	yes	0.2	12.8	20.1
accelerations	$\mathcal{O}(n^2)$	yes	2.8	10.0	14.2
other	$\mathcal{O}(n)$	attempted[d]	0.3	7.1	14.2
total active[c] communication			N/A	16.7	28.2

[a] The operations were timed using TAU (http://www.cs.uoregon.edu/research/tau/home.php). [b] Includes the active communication and synchronization time due to load imbalance. [c] Does not include synchronization time due to load imbalance. [d] The corresponding code was parallelized, but communication overhead turned out to be greater in the present implementation than the gain from splitting the workload.

are the correlation terms; however, this is because we introduced a 7 Å distance cutoff on the computation of these interactions.[29] The single-body terms [of $\mathcal{O}(n)$ complexity] are by far less expensive. It can also be seen that, although energy and gradient computations constitute the dominant fraction of computation time with one processor, gradient transformation and computing accelerations from forces are operations of $\mathcal{O}(n^2)$ complexity and cannot be ignored in parallelizing the code. Moreover, the complexity of these operations cannot be reduced by introducing the cutoff.

2.5.3. Fine-Grained Parallelization of Energy- and Gradient-Component Calculations. The following three approaches are commonly implemented in the parallelization of energy and force calculations:[15,63,64] the particle- (atom)-decomposition approach also called the replicated-data approach, the domain- (spatial)-decomposition approach, and the force-decomposition approach.

In the particle-decomposition approach, N atoms are split between P processors, and each of the processors calculates the forces acting at the atoms assigned to it. The coordinates of all atoms must be known to all processors. The particle decomposition approach is naturally load-balanced if no cutoff is introduced on nonbonded interactions; when a cutoff is introduced, load-balancing can be achieved by randomizing the order of the atoms before dividing them between processors.[63,64] However, the memory requirements and cost of communication are on the order of $\mathcal{O}(N)$ (i.e., does not decrease with increasing number of processors). The all-to-all communication is required because a given processor must obtain the updated coordinates from all other processors. Moreover, because Newton's third law is usually implemented to reduce the amount of computations, the forces are exchanged between the processors, which requires additional communication.

In the domain-decomposition approach, the space occupied by the system is partitioned between processors, and each processor handles the particles which are contained in the region of space assigned to it. The load balance is easy to maintain only when the space is nearly uniformly filled with particles, as in all-atom simulations with explicit solvent carried out in a periodic solvent box; then each processor always handles approximately the same number of particles.[15,63,64] It is difficult to maintain the load balance for irregularly shaped systems, typical of simulations of proteins with implicit solvent, for which particle density varies significantly. The domain-decomposition algorithm implies the least memory requirement and communication cost; both are on the order of $\mathcal{O}(N/P)$. The all-to-all communication is avoided here because a given processor needs only the information from the processors that are assigned to the neighboring regions of space.

In the force-decomposition algorithm, the interatomic interactions are partitioned between processors in such a way that a given processor owns a block of force matrix pertaining to forces due to atoms from, say, i to j acting on atoms from, say, k to l.[63,64] Thus, this processor needs the coordinates of the atoms of only these two groups and the all-to-all communication is avoided. Consequently, the memory requirement and communication cost are on the order of $\mathcal{O}(N/\sqrt{P})$, i.e., between that of the particle- and domain-decomposition algorithms. As in the particle-decomposition algorithm, the load balance is natural with no cutoff on nonbonded interactions; with a cutoff, it can be maintained by randomizing the distribution of forces between the processors.

The UNRES energy terms of eq 1 and the corresponding forces can be divided into the following classes:

(1) The terms computed independently from the coordinates of UNRES objects within

(a) a single residue (U_{bond}; one-body terms)

(b) a number (up to 4) of neighboring residues (U_b, U_{rot}, U_{tor}, U_{tord}, $U_{turn}^{(3)}$, $U_{turn}^{(4)}$; local terms)

(c) pairs of residues ($U_{SC_iSC_j}$, $U_{SC_ip_j}$, $U_{p_ip_j}^{el}$, $U_{p_ip_j}^{VDW}$; pairwise terms)

(d) pairs of residues plus those of neighboring residue ($U^{(3)}_{\rm corr}$; pairwise correlation terms)

(2) The terms which comprise two neighboring pairs of interacting peptide groups p_ip_j and $p_{i+1}p_{j\pm1}$ ($U^{(4)}_{\rm corr}$, $U^{(5)}_{\rm corr}$, $U^{(6)}_{\rm corr}$, $U^{(6)}_{\rm turn}$).

As follows from Table 1, the bulk effort involves the computation of the pairwise terms included in 1c and 1d. Therefore, we designed energy and force parallelization to distribute these terms in the first place. In the present UNRES, no cutoff is applied on nonbonded interactions. Consequently, the workload corresponding to the computation of terms 1a−1d remains fixed, and each of these terms can be distributed to processors at the beginning of the calculations. It appears natural to use the particle- or force-decomposition approach. Given the fact that the solvent is implicit in UNRES and, therefore, particle density is not uniform, using the domain-decomposition approach would require a sophisticated algorithm to partition the space between processors (such as those designed for, e.g., solving the boundary-value problems with irregular shapes).[65] Moreover, the equations of motion are expressed in virtual-bond vectors in UNRES/MD, and therefore, communication with non-neighboring boxes could not be avoided because the forces expressed in UNRES interaction-site coordinates must be transformed to those expressed in virtual-bond vectors. Furthermore, the number of interaction sites is greatly reduced in UNRES compared to all-atom MD, and neither does storing all coordinates at each processor pose memory problems nor does sending the updated coordinates to all processors pose a significant communication problem.

Given the above considerations, we designed a parallelization scheme which is similar to force-decomposition algorithms in that the interactions and non-interaction sites are split between the processors, although we partition the upper triangular of the force matrix and not the whole force matrix. Thus, we make explicit use of Newton's third law. The decomposition of pairwise interactions is illustrated in Figure 4. Communication is not required to update the interaction list assigned to each processor. On the other hand, our scheme is similar to the particle-decomposition scheme in that the coordinates (virtual-bond vectors) are sent to all processors (Figure 2b). However, because only the FG master processor updates the virtual-bond vectors, this involves broadcast and not all-to-all operations.

The computation of the interactions included in group 1 was parallelized in the same manner; it should be noted, however, that the list of interactions consigned to a processor remains constant independent of the use of a cutoff. Although the complexity of the local interactions is only $\mathcal{O}(n)$, parallelizing this part of the code does not introduce any extra communication and, consequently, was implemented in UNRES.

Group 2 contains neighboring pairs of peptide groups, and quantities calculated in computing $U^{\rm el}_{p_ip_j}$ and $U^{\rm el}_{p_{i+1}p_{j\pm1}}$ are subsequently utilized in computing the correlation terms involving the respective pairs of peptide groups.[29,30] A pair of interactions necessary to compute a contribution to one of the UNRES energy terms listed in point 1d can happen to be split between two processors, and communication is,

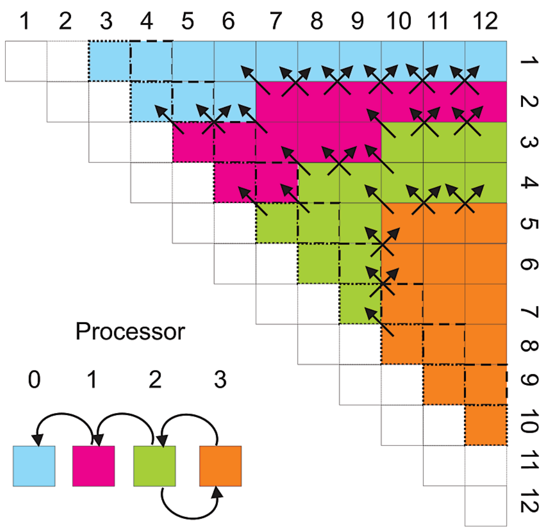


Figure 4. Illustration of domain decomposition of peptide-group interactions of a 13-residue chain (with 12 peptide groups) and exchange of interaction information to compute correlation interactions. A box in the upper-triangular array shown in the upper part of the graph corresponds to the interaction between peptide groups numbered by the row and the column of the array. Different colors mark assignment of the respective interactions to different processors, whose ranks are shown in the lower small panel. The white diagonal and next to diagonal boxes correspond to the same or neighboring peptide groups, for which long-range interactions are not considered. The dot-dashed and dashed lines lines border the 1,3 and 1,4 interactions, respectively. The arrows in the upper panel indicate the directions in which the interaction information is transmitted, and the half-circular arrows in the left lower panel show the flow of interaction information.

therefore, required to avoid multiple computation of the same quantities by different processors.

Because the quantities required to compute $U^{\rm el}_{p_ip_{i+2}}$ and $U^{(3)}_{{\rm corr};\ p_ip_{i+2}}$ are also required for $U^{(3)}_{{\rm turn};\ i}$, and those required to compute $U^{\rm el}_{p_ip_{i+3}}$ and $U^{(3)}_{{\rm corr};\ p_ip_{i+3}}$ are required to compute $U^{(4)}_{{\rm turn};\ i}$, computations of $U^{\rm el}_{p_ip_{i+2}}$ and $U^{(3)}_{{\rm corr};\ p_ip_{i+2}}$ interactions, respectively, as well as those of $U^{\rm el}_{p_ip_{i+3}}$ and $U^{(3)}_{{\rm corr};\ p_ip_{i+3}}$, respectively, are handled by separate loops over the p_i, p_{i+2} and p_i, p_{i+3} pairs, respectively, to provide better load balance. The long-range electrostatic and correlation interactions are computed from the p_i, p_{i+4} pairs, onward. The loops to compute interactions are then distributed to the processors assigned to a CG task, subject to optimum load balancing. As mentioned earlier in this section, no cutoff is applied to the interactions listed in points 1c or 1d, and consequently, the lists of these interactions consigned to each fine-grained processor are fixed. For each energy term listed in points 1a and 1b, each fine-grained processor is assigned the range of the residues (in terms of the number of the residue to start and to end at) to compute this term. For the terms listed in points 1c and 1d, each processor is assigned a list of the start and end residue numbers to process for each first pair index (i). If the first element of entry i of the list of interactions that belongs to processor i is zero, this processor does not have any interaction such that residue i is the first in the pair to process. The number of components of each of the energy terms that belongs to the 1a, 1b, or 1c category is optimally load-balanced for each processor and differs by 1 at most for different processors. For the interactions between the peptide groups, interaction distribution between processors (along with the assignment of the pertinent correlation interactions) is illustrated in Figure 4.

For the interactions listed in point 2, a 7 Å cutoff is applied to the distances between peptide groups ($r_{p_ip_j}$ and $r_{p_ip_{j\pm1}}$) involved in a correlation, with a contact function given by eq 42 of ref 29, which varies smoothly with the distance from 1 to 0 from $r = 6.9$ to $r = 7$ Å. Therefore, a list of interactions to be sent by a given processor to other processors and received by it from other processors is dynamic.

Before calculations are actually started, each processor (let its rank be R) creates the list of processors to which quantities pertaining to each pair of peptide groups might need to be sent and a list of processors from which it can receive terms pertaining to pairwise interactions. The algorithm to construct these lists is as follows:

Send List

(1) Loop through the p_ip_{i+2} pairs consigned to the processor. For each pair, loop through the other processors with ranks less than R, and add processor S to the send list of p_ip_{i+2} interaction at processor R if processor S has (i) the $p_{i-1}p_{i+1}$ interaction or (ii) the $p_{i-1}p_{i+3}$ interaction.

(2) Loop through the p_ip_{i+3} pairs consigned to the processor. For each pair, loop through the other processors with ranks less than R and add processor S to the send list of p_ip_{i+3} interaction at processor R if processor S has (i) the $p_{i-1}p_{i+2}$ interaction or (ii) the $p_{i-1}p_{i+4}$ interaction.

(3) Loop through all p_ip_j pairs consigned to the processor such that $j > i + 3$. For each pair, loop through the other processors with ranks less than R and add processor S to the send list of p_ip_j interaction at processor R if processor S has (i) the $p_{i-1}p_{j-1}$ interaction or (ii) the $p_{i-1}p_{j+1}$ interaction or (iii) the $p_{i+1}p_{j-1}$ interaction. Situation iii occurs only if $p_{i+1}p_{j-1}$ is a pair of 1,3- or 1,4-adjacent peptide groups (i.e., $j - i = 4$ or $j - i = 5$), because these interactions are distributed to processors independent of the longer-range interactions, and consequently, a processor with a rank lower than R might have a 1,3 or a 1,4 interaction with greater i than the long-range interactions consigned to processor R.

Receive List

The construction of the receive list of a receiving processor R is the reverse of the construction of the send list. Therefore, processors with ranks greater than R are searched in steps 1 and 3 by the receiving processor R, and if a processor is found to contain interactions that might be needed by R, its rank is added to the receive list of processor R. The ranks of the processors which R can potentially receive interactions from is the only information stored in the receive list, because the actual interaction list to be received from a given processor depends on conformation. The interaction list is, therefore, provided to R every time the correlation interactions consigned to this processor are calculated.

For processor R, each pair of peptide groups from the preset send lists is checked, and if the contact function value is greater than 0, all information pertaining to this peptide group pair, needed to compute correlation interactions and their gradients, is transferred to buffers intended for processors of the send list of this interaction. After the entire preset send list has been scanned, nonblocking send operations to each of the processors from the send list are initiated. If there are no interactions to be sent to processor S, the number 0 is sent to indicate this fact. After the send operations have been initiated, nonblocking receive operations are executed. A waitall operation completes the send−receive process to synchronize all fine-grained tasks. Nonblocking send and receive operations are used to exchange interaction information with a wait operation to synchronize the calculation of correlation energy. A scheme of information exchange is presented in Figure 4.

Each processor adds the received interactions to the list of its own computed interactions; however, it marks them as "received" to prevent double-counting of the respective correlation interactions. The latter could happen if, e.g., the processor has the interaction p_kp_l and needs to receive $p_{k+1}p_{l+1}$ and, at the same time, has $p_{k+1}p_{l-1}$ and needs to receive $p_{k+2}p_l$ to compute the respective correlation interactions. The received interactions $p_{k+1}p_{l+1}$ and $p_{k+2}p_l$ themselves form a correlation interaction (since a correlation interaction is formed both by p_ip_j and $p_{i+1}p_{j+1}$ and by p_ip_j and $p_{i+1}p_{j-1}$).[29] Consequently, if they were not marked "received", this additional interaction would be computed by the processor that received both of them, and since this interaction is computed by the processor which owns $p_{k+1}p_{l+1}$, the interaction would be doubly counted. It should be noted that marking the received interactions does not imply any additional communication. A correlation term pertaining to p_kp_l and $p_kp_{l\pm1}$ is not computed if both pairs of interactions are marked "received". This is because a given processor always has one interaction of its own of a pair of interactions that constitute a correlation term (see Figure 4).

As described above, the distribution of correlation interactions between processors provides a nearly optimal load balance, given the fact that the interactions between peptide groups are optimally distributed between the fine-grained processors. If the 7 Å cutoff, which is imposed on each of the pairs of interacting residues to compute the respective correlation interaction,[29] is not in effect (for small proteins), the number of correlation terms is twice the number of interactions consigned to the processor except when it has the first or the last residue. For medium-sized and large proteins, the cutoff which decreases the number of correlations is in effect; however, even in this case, the number of correlation interactions is nearly the same for each fine-grained processor, because the number of peptide groups within the cutoff distance from a given peptide group remains proportional to the total number of peptide groups with which it interacts.

2.5.4. Fine-Grained Parallelization of Other Operations. Computing accelerations from forces in UNRES involves multiplication of the vector of forces expressed in virtual-bond vectors by the inverse of the matrix G [eq 12]. Since the part of the forces is expressed in internal coordinates or in Cartesian coordinates of the UNRES interaction sites, these forces first need to be transformed to the space of virtual-bond vectors. U_b, U_{tor}, and U_{tord}, as well as the correlation terms except $U_{corr}^{(4)}$ are expressed in backbone virtual-bond (θ) and virtual-bond-dihedral (γ) angles. The transformation takes the following form:

$$\nabla'_{\mathbf{dC}_i}U_X = \frac{\partial U_X}{\partial\theta_{i-1}}\nabla_{\mathbf{dC}_i}\theta_{i-1} + \frac{\partial U_X}{\partial\theta_i}\nabla_{\mathbf{dC}_i}\theta_i + \frac{\partial U_X}{\partial\gamma_{i-2}}\nabla_{\mathbf{dC}_i}\gamma_{i-2} + \frac{\partial U_X}{\partial\gamma_{i-1}}\nabla_{\mathbf{dC}_i}\gamma_{i-1} + \frac{\partial U_X}{\partial\gamma_i}\nabla_{\mathbf{dC}_i}\gamma_i \quad (14)$$

where U_X is U_b, U_{tor} or U_{tord}, $U_{corr}^{(3)}$, $U_{corr}^{(5)}$, $U_{corr}^{(6)}$, U_{turn}^{3}, $U_{turn}^{(4)}$, and $U_{turn}^{(6)}$ and $\nabla'_{\mathbf{dC}}$ indicates that we consider only the part of the derivatives corresponding to dependence on the θ and γ angles (it should be noted that only U_b depends on θ angles). Therefore, the transformation of this part of the gradient involves an $\mathcal{O}(n)$ effort, where n is the number of residues and can be predicted not to benefit from parallelization. Nevertheless, we included an option to fine-grain the operations given by eq 14, but communication overhead was greater than the profit from fine-graining, except for a small number of processors. The rotamer potentials (U_{rot}) and the virtual-bond-deformation potentials (U_{bond}) are expressed directly as functions of virtual-bond vectors,[38] and consequently, no force transformation is needed for these terms.

The derivatives of both the pairwise and correlation terms in **q** are expressed by eq 15.

$$\nabla_{\mathbf{q}}U_X = \mathbf{A}^{\mathrm{T}}\nabla_{\mathbf{x}}U_X \quad (15)$$

where $\nabla_{\mathbf{y}} = (\partial/\partial y_1, \partial/\partial y_2, ..., \partial/\partial y_n)$ is the gradient operator. Recalling eq 6, it can be realized that the transformation of the energy gradient in site positions (**x**) to that in the virtual-bond vectors forming the vector of generalized coordinates **q** [eq 3] involves only summations [the effort being roughly $\mathcal{O}(n^2)$] and divisions by 2 [the effort being $\mathcal{O}(n)$]. This observation was implemented in writing the code. Nevertheless, the $\mathcal{O}(n^2)$ effort requires parallelizing this part of the computations.

The matrix **A** and the vector $\nabla_{\mathbf{x}}U_X$, as well as the matrix **G** and vector **F**, are divided between processors, as shown in Figure 5. Each processor executes its part of the computations, and reduction to the master fine-grain task with the sum operator (in the MPI terms) is executed at the end.

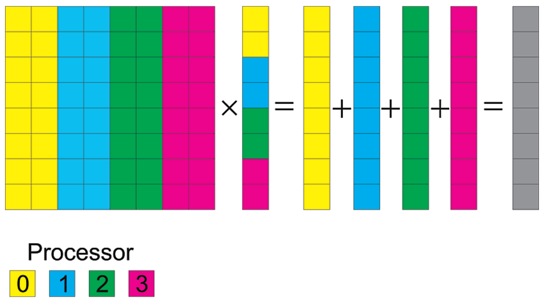


Figure 5. Partitioning of the matrix **A** and vector $\nabla_{\mathbf{x}}U$(a) and of the matrix **G** and the vector $\nabla_{\mathbf{q}}$ between the fine-grain tasks for distributed computing of $\nabla_{\mathbf{q}}$ and accelerations, respectively. Different colors mark different processors (rank shown in the small lower panel). The matrix−vector multiplication is distributed between processors (each handling only its part of the matrix and of the vector) to give the vectors containing incomplete sums in each component. The vectors are subsequently summed up to give the resulting vector.

3. Results and Discussion

3.1. Fine-Graining a Single MD Trajectory. To assess the performance of the fine-grained part of the code, we carried out short canonical MD simulations with the Berendsen thermostat[57] implemented in UNRES[42] for the following nine proteins with size from 37 to 767 residues: 1E0L (37 residues), 1BDD (46 residues), 1KOY (62 residues), 2K4N (111 residues), 2K5I (154 residues), 1P1D (196 residues), 3G5A (304 residues), 2KHO (600 residues), and 1TF5 (767 residues). We used the recent force field calibrated by extensive random exploration of energy-parameter space.[56] The tests were run on the machines listed in section 2.4. From 100 to 10 000 UNRES MD steps, depending on protein size and processor speed, with 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, and 1024 processors, respectively, were taken; the largest number of processors was tested only on the machines with the fastest communication (jugene.fz-juelich.de and intrepid.alcf.anl.gov). The number of MD steps was the same for a given protein and machine (the strong-scaling runs).

The nonsetup time (defined as the wall-clock time used to perform MD steps) was taken to assess parallel performance. The setup time, not taken into consideration, includes data reading, computing and inverting the inertia tensor, and generating initial velocities. For production runs, the setup time is negligible. The nonsetup times per MD step for the above-mentioned proteins, each run with a single processor of the machines listed above, are plotted in Figure 6a; additionally, the per-day simulation lengths (in nanoseconds/day), together with the per-day lengths corresponding to the all-atom GROMACS 4.0.5 program run on jugene.fz-juelich.de for 1BDD, 1P1D, and 1TF5, are plotted in Figure 6b. As can be seen from this figure, in all cases the single-processor nonsetup time grows with the square of the number of residues (a slope of about 2 in all four plots of Figure 6), as expected because most of the computational effort is $\mathcal{O}(n^2)$, where n is the number of residues in a polypeptide chain. For a given protein, the non-setup times increase in the order galera.task.gda.pl < bigben.psc.edu < jugene.fz-juelich.de < intrepid.alcf.anl.gov, which conforms to the processor speed of these machines. With cutoff introduction on nonbonded interactions, the GROMACS computation time scales almost linearly with protein size.

In Figure 7a−d and Figure 8a−d the speedups and efficiencies, respectively, for the nine proteins obtained with galera.task.gda.pl, bigben.psc.edu, intrepid.alcf.anl.gov, and jugene.fz-juelich.de, respectively, are plotted as functions of the numbers of processors. The speedup (s) and efficiency (η) are calculated from eqs 16 and 17, respectively.

$$s(P) = \frac{t(1)}{t(P)} \quad (16)$$

$$\eta(P) = \frac{t(1)}{Pt(P)} \tag{17}$$

where P is the number of processors, $t(1)$ is the time of a single-processor run, and $t(P)$ is the time of a run with P processors.

It can be seen that the best scalability is achieved with both IBM BlueGene/P supercomputers (a speedup over 4 is achieved even for the smallest protein 1E0L), while galera.task.gda.pl represents the poorest parallel performance except for runs for the smallest proteins with up to 4 processors (in which case the quad-core architecture of this computer is an advantage) for which it has better performance than bigben.psc.edu. These observations conform to the difference in the computation and communication speed of the four supercomputers mentioned. As an example, the communication times pertaining to collective operations (MPI_REDUCE, MPI_GATHER, and MPI_SCATTER) expressed relative to single-processor time and relative to the wall-clock time of a given multiprocessor run are plotted in Figure 9a and b, respectively, for the largest protein (1TF5; 767 residues) as function of the number of processors. As can be seen from Figure 9, the ratio of the communication time is the smallest for both IBM BlueGene/P machines; moreover, it varies little between 8 and 128 processors. With machines with not so fast communication, the communication time constitutes even 70% of wall-clock time. Because of a nearly constant communication-time overhead, the speedup and efficiency obtained from the runs on BlueGene/P computers for larger proteins quite strictly obey the classical Amdahl law [eq 18].[66]

$$s(P) = \left[f + \frac{1-f}{P}\right]^{-1} \tag{18}$$

where s is the speedup, P is the number of the processors, and f is the fraction of time spent when running the nonparallelized part of the code with one processor. The dependence of the communication time on the number of processors exhibits the largest slope for galera.task.gda.pl.

A more detailed analysis of the timing is provided in Table 1 with the example of 1TF5 (the largest protein considered in our study) run on bigben.psc.edu. It can be seen that, although the nonparallelized operations (such as calculations of virtual-bond angles and virtual-bond dihedral angles and calculation of auxiliary quantities to compute $U_{\text{corr}}^{(3)}$ and $U_{\text{turn}}^{(3)}$ and $U_{\text{turn}}^{(4)}$) constitute a negligible fraction of the CPU time with a single processor, they take relatively more and more time with an increasing number of processors and, consequently, reduce the available speedup according to Amdahl's law and gradually become the bottleneck of the computations. As pointed out in section 2.5.4, we attempted to parallelize these operations, but the communication overhead turned out to be greater than the gain from eliminating this nonparallelized component of computations. However, it is possible that the scalability of UNRES calculations can benefit from parallelizing the above-mentioned operations on mixed shared-and-distributed memory architectures. Additionally, the calculation of $U_{\text{corr}}^{(4)}$ is not fully load-balanced at present (see section 2.5.3) and, consequently, does not scale very well (see Table 1); however, the relative contribution of the computation of $U_{\text{corr}}^{(4)}$ to the total computation time is small. Nevertheless, a better parallelization of this energy component is necessary for further extension of the time scale of UNRES calculations. The load imbalance can be addressed in the force-decomposition scheme by randomizing distribution of peptide groups on processors in a single preprocessing step.[63,64]

A practical issue in molecular simulations is the wall-clock time in which simulations can be accomplished. In Figure 10a, the minimum nonsetup times per MD (irrespective of efficiency of a parallel run) and, in Figure 10b, the nonsetup times at 50% efficiency are plotted vs the number of residues in a chain. Additionally, the data corresponding to 50% efficiency are presented in Figure 10c as the achievable length of simulation expressed in nanoseconds of MD

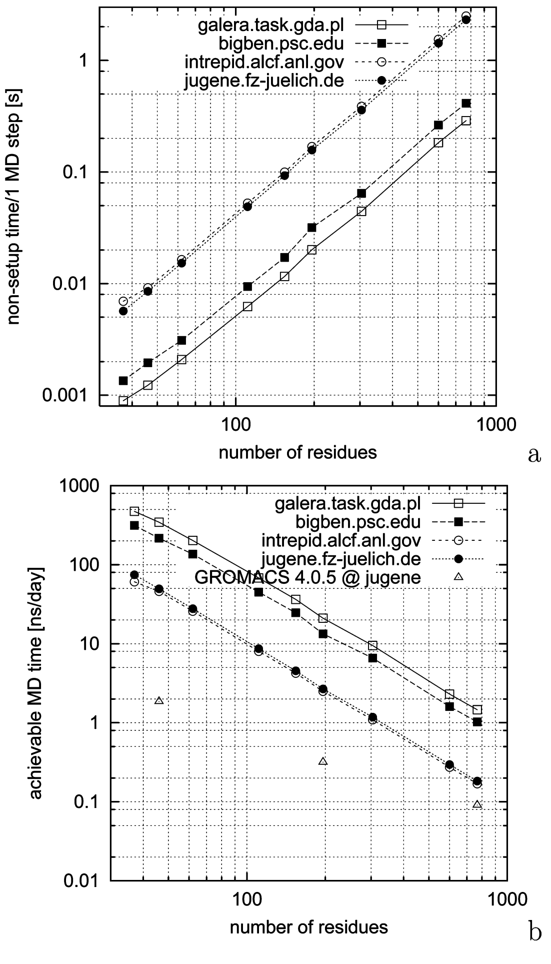


Figure 6. Plots of (a) nonsetup times per MD step and (b) the achievable simulation length (ns/day) as functions of number of residues with a single processor for the four supercomputers used in this study. A logarithmic scale is used on both axes; it can be seen that the slope is approximately 2, indicating a quadratic dependence on the number of residues. The per-day simulation lengths achievable with GROMACS 4.0.5 at jugene.fz-juelich.de are also included for comparison. The nonsetup time per MD step (in seconds) can be expressed approximately as $t = 4.95 \times 10^{-7}n^2$ for galera.task.gda.pl, $t = 7.0 \times 10^{-7}n^2$ for bigben.psc.edu, $t = 4.26 \times 10^{-6}n^2$ for intrepid.alcf.anl.gov, and $t = 3.94 \times 10^{-6}n^2$ for jugene.fz-juelich.de, respectively, where n is the number of residues.

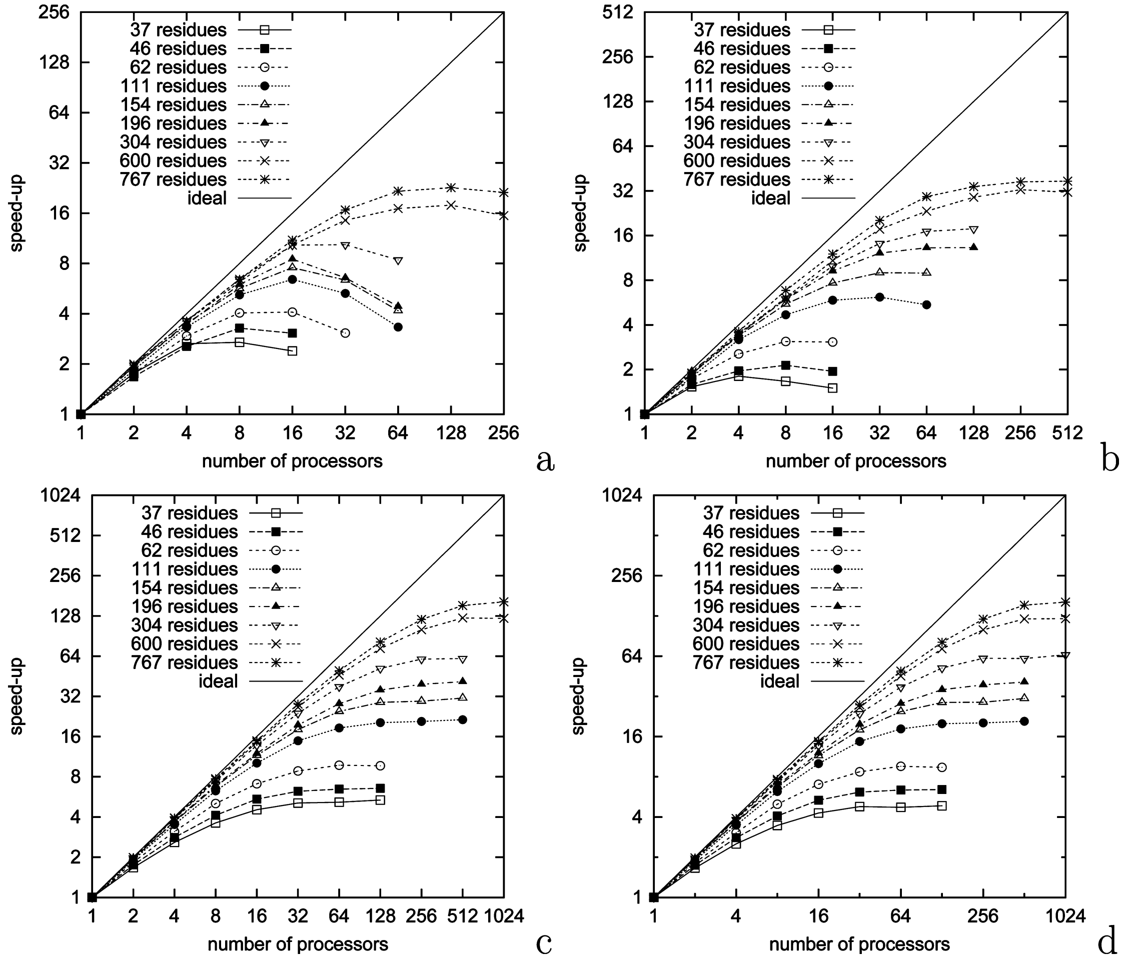


Figure 7. Plots of the speedups for canonical single-trajectory UNRES/MD runs obtained for proteins with various numbers of residues vs the number of processors on (a) galera.task.gda.pl, (b) bigben.psc.edu, (c) intrepid.alcf.anl.gov, and (d) jugene.fz-juelich.de. The logarithmic scale with base 2 is used on both axes.

simulation per day, which is a commonly used metric.[15] It can be seen that the curves presented in Figure 10a and b are slightly convex initially, then approximately linear, and for the largest proteins, the slope starts to decrease. The convex shape of the initial portion of the time vs number of residues plots is the most pronounced for galera.task.gda.pl, while for IBM BlueGene/P supercomputers, the dependence of the minimum nonsetup time and the nonsetup time at 50% efficiency on the number of residues is nearly linear. The approximately linear dependence of the wall-clock time per MD step on protein size is a big advantage over single-processor runs in which this dependence is quadratic (Figure 6). On the other hand, it can be seen from Figure 10 that fast communication does not overcome greater processor speed. The lines corresponding to both IBM BlueGene/P supercomputers are always above those of the other two machines with faster processors. For the 1TF5 protein (the largest one), the minimum and the 50% efficiency times are longer by about 12% and 30% on IBM BlueGene/P than on galera.task.gda.pl and bigben.psc.edu, respectively. On the other hand, this is still better than the ratio of the single-processor time of the faster IBM BlueGene/P (jugene.fz-juelich.de) to that of galera.task.gda.pl and bigben.psc.edu, which are 6.36 and 4.21, respectively (however, 4 times more processors have to be used on IBM BlueGene/P).

For reference, the single-processor nonsetup times of the small 1KOY protein (62 residues) are also indicated in Figure 10. For this small protein, 20 000 000 MD steps (100 ns with the 5 fs time step used in UNRES simulations), which is required for a converged canonical or replica-exchange UNRES/MD run, can be accomplished in about 12 h of wall-clock time on galera.task.gda.pl and in about 18 h on bigben.psc.edu, respectively, without fine-graining the code. It can be seen from Figure 10 that fine-graining enables us to accomplish simulations for about 200-residue proteins on galera.task.gda.pl and bigben.psc.edu in the same wall-clock time as for 1KOY. For the two IBM BlueGene/P computers, simulations of the largest (1TF5) protein can be accomplished in a shorter wall-clock time than that required for 1KOY with these machines, if efficiency is neglected (Figure 10a), or in a slightly greater wall-clock time with 50% efficiency (Figure 10b). With the maximum speedup achievable, 20 000 000 MD-step simulations for 1TF5 can be accomplished in about 80 wall-clock hours (3.3 wall-clock days). A millisecond of simulations for this protein (effectively a second, given the ~1000-times lengthening of the UNRES time scale with respect to the experimental time scale)[42,43] would require only slightly longer than a month of computations on IBM BlueGene/P. This achieve-

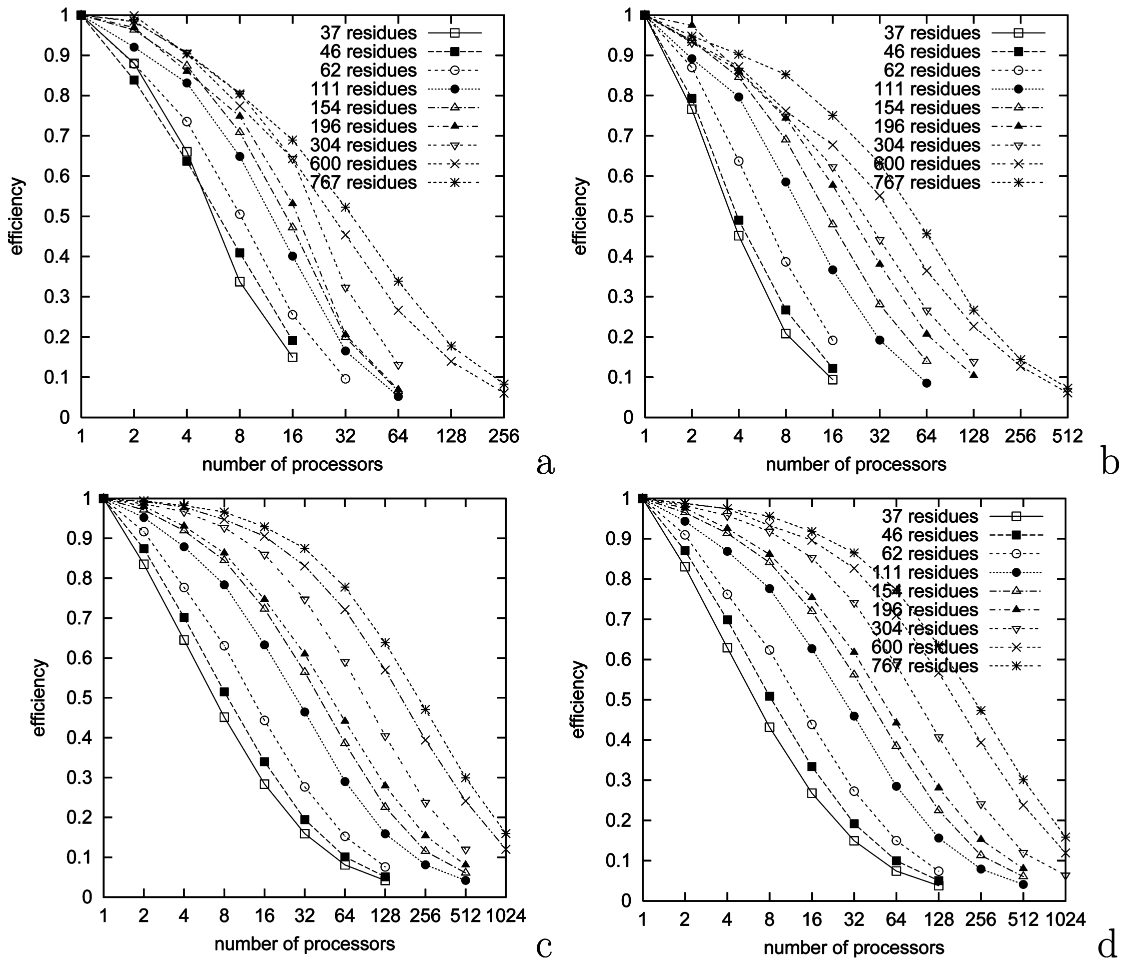


Figure 8. Plots of the efficiencies for canonical single-trajectory UNRES/MD runs obtained for proteins with various numbers of residues vs the number of processors on (a) galera.task.gda.pl, (b) bigben.psc.edu, (c) intrepid.alcf.anl.gov, and (d) jugene.fz-juelich.de. The logarithmic scale with base 2 is used on the abscissae.

ment makes the biological time scales achievable with the UNRES model.

In Figure 10c, the data corresponding to all-atom calculations with GROMACS[15] at 50% efficiency with jugene.fz-juelich.de are also presented. GROMACS is currently one of the most efficient and best load-balanced MD software programs. An 11 Å cutoff was applied on all nonbonded interactions. As shown, UNRES provides 5−7 times longer per-day simulation length compared to GROMACS. Because the UNRES event-based time scale is at least 4 times longer than that of the all-atom approach,[42,43] the effective speedup with respect to GROMACS is at least 20 times. When the single-processor times are compared, the UNRES per-day simulation length is 27 times greater for 1BDD, 8 times greater for 1P1D, and 2 times greater for 1TF5 (see also Figure 6b). The decrease of the UNRES/GROMACS per-day simulation length with a single processor is not surprising, because UNRES does not implement a cutoff on nonbonded interactions, as opposed to GROMACS and, consequently, the complexity of UNRES is $\mathcal{O}(n^2)$, while that of GROMACS (which implements the cutoff) is $\mathcal{O}(n)$. Therefore, introducing a cutoff on nonbonded interactions in UNRES and, thereby, reducing the complexity to $\mathcal{O}(n)$, could be advantageous for large proteins. Preliminary results with an 11 Å cutoff on nonbonded interactions in UNRES have shown that the scaling becomes almost linear and the ratio of single-processor per-day simulation time with UNRES to that with GROMACS becomes 10 for 1TF5.

3.2. Two-Grain Multiplexed Replica-Exchange Simulations. The scalability data reported in section 3.1 pertain to single-trajectory runs or canonical runs with multiple independent trajectories. In the replica-exchange (REMD) and multiplexed replica exchange molecular dynamics (MREMD) simulations, exchange of information occurs. Fine graining might influence the scalability of the coarse-grained tasks because even residual load imbalance can cause remarkable differences in the wall-clock time required to compute the number of MD steps between exchanges. In this section, we, therefore, report the results of the scalability of short MREMD runs. We chose only three proteins: 1KOY (small size), 1P1D (medium size), and 2KHO (large size). The calculations were run on galera.task.gda.pl, bigben.psc.edu, and jugene.fz-juelich.de; we omitted intrepid.alcf.anl.gov because it gives the same scalability profiles as those of jugene.fz-juelich.de. We ran from 1 to 256 MREMD trajectories (the numbers of trajectories were the consecutive powers of 2 and simulations for 1 trajectory were taken as reference because they were canonical MD simulations). The number of fine-grained processors was generally either 1 (reference for an MREMD simulation with a given number

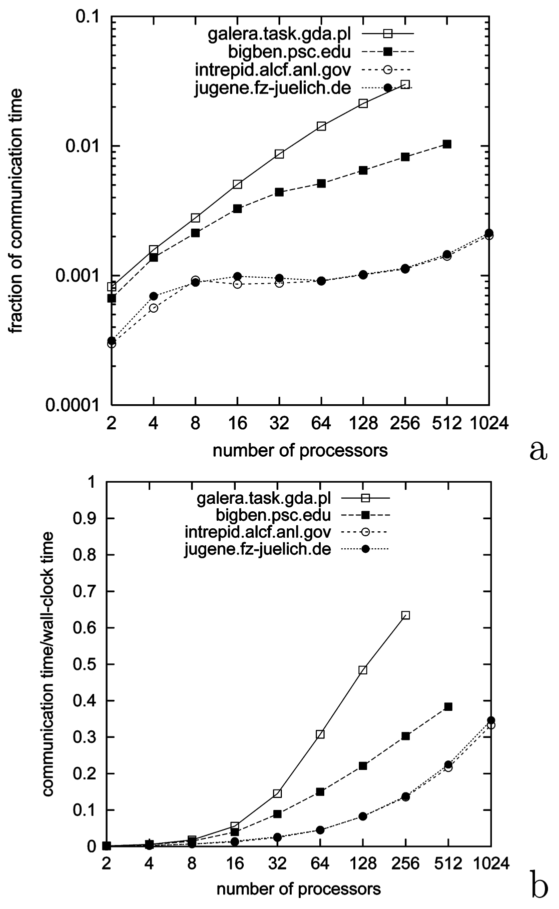


Figure 9. Plots of (a) the fraction of the collective-communication time to single-processor nonsetup time vs the number of processors in a single-trajectory UNRES/MD run for 1TF5 and (b) fraction of the collective-communication time to the wall-clock time with a given number of processors. Logarithmic scales are used on both axes.

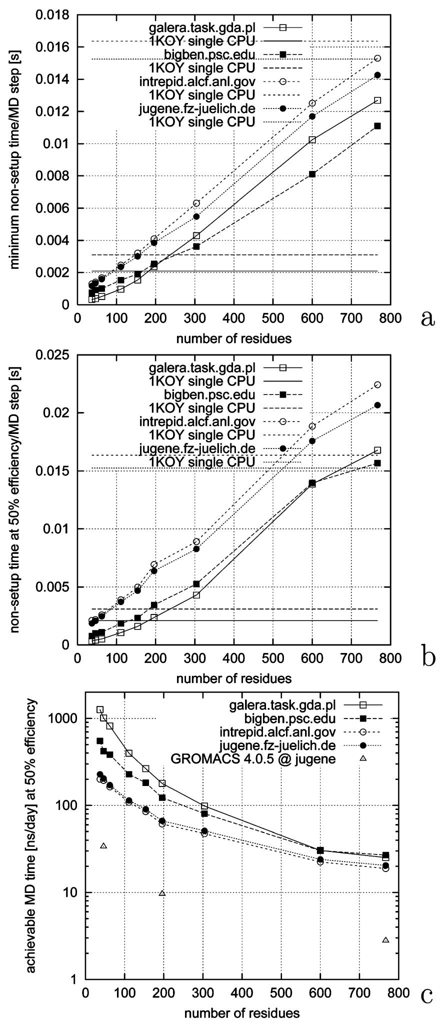


Figure 10. Plots of the minimum nonsetup time per MD step (a), the nonsetup time per MD step with 50% parallel efficiency (b), and the achievable formal simulation length (in nanoseconds/day; taking the 4.89 fs MD time step) with 50% efficiency (c) vs the number of residues obtained in fine-grained single-trajectory UNRES/MD runs on galera.task.gda.pl, bigben.psc.edu, intrepid.alcf.anl.gov, and jugene.fz-juelich.de. For references, horizontal lines (solid for galera.task.gda.pl, long-dashed for bigben.psc.edu, short-dashed for intrepid.alcf.anl.gov, and dotted for jugene.fz-juelich.de) are drawn which correspond to single-processor nonsetup times of the 62-residue 1KOY protein. The time at 50% efficiency was obtained by linear interpolation from the times at the efficiencies bracketing 50%. The achievable simulation lengths at 50% efficiency obtained for 1BDD, 1P1D, and 1TF5 proteins with GROMACS 4.0.5[15] (taking the 2 fs time step) run at jugene.fz-juelich.de are also included in part c for comparison. For 1E0L, 1BDD, 1KOY, 2K4N, 2K5I, 1P1D, 3G5A, 2KHO, and 1TF5, respectively, the numbers of processors corresponding to part a are 8, 8, 16, 16, 16, 16, 32, 128, and 128 with galera.task.gda.pl; 4, 8, 8, 32, 32, 128, 128, 256, and 512 with bigben.psc.edu; and 128, 128, 64, 128, 128, 512, 1024, 1024, and 1024 with intrepid.alcf.anl.gov and jugene.fz-juelich.de. The numbers of processors corresponding to UNRES runs reported in parts b and c are 8, 8, 16, 16, 16, 16, 32, 128, and 128 with galera.task.gda.pl; 4, 8, 8, 32, 32, 128, 128, 256, and 512 with bigben.psc.edu; and 8, 8, 16, 32, 32, 64, 64, 128, and 256 with intrepid.alcf.anl.gov and jugene.fz-juelich.de. The numbers of processors used in the GROMACS runs reported in part c were 32 for 1BDD and 64 for 1P1D and 1TF5.

of trajectories) or corresponding to about 50% efficiency for a given protein and a given machine (section 3.1). However, in runs on the IBM BlueGene/P machine, we also used the number of fine-grained processors corresponding to nearly maximum scalability. The number of MD steps per trajectory and the frequency of replica exchange were the same for a given protein (weak-scaling tests); 40 000 steps with exchange each 10 000 steps for 1KOY, 20 000 steps with exchange each 5000 steps for 1P1D, and 10 000 steps with exchange each 5000 steps for 2KHO, respectively. The most massively parallel calculation was that run for 2KHO on jugene.fz-juelich.de; it comprised 256 trajectories (CG tasks), each run with 256 processors (FG tasks), i.e., 65 536 processors total.

Plots of the nonsetup time and efficiency vs the number of trajectories (nonfine grained tasks) for different numbers of fine-grain tasks are shown in Figures 11, 12, and 13 for 1KOY, 1P1D, and 2KHO, respectively. It can be seen that the efficiency of MREMD calculations decreases slightly with the number of trajectories, although the decrease is not remarkable. The most remarkable decrease is observed for the runs on the IBM BlueGene/P supercomputer; there is a clear efficiency drop (about 5%), after which the efficiency does not decrease remarkably with the number of trajectories. It can be noted that, except for the smallest 1KOY protein,

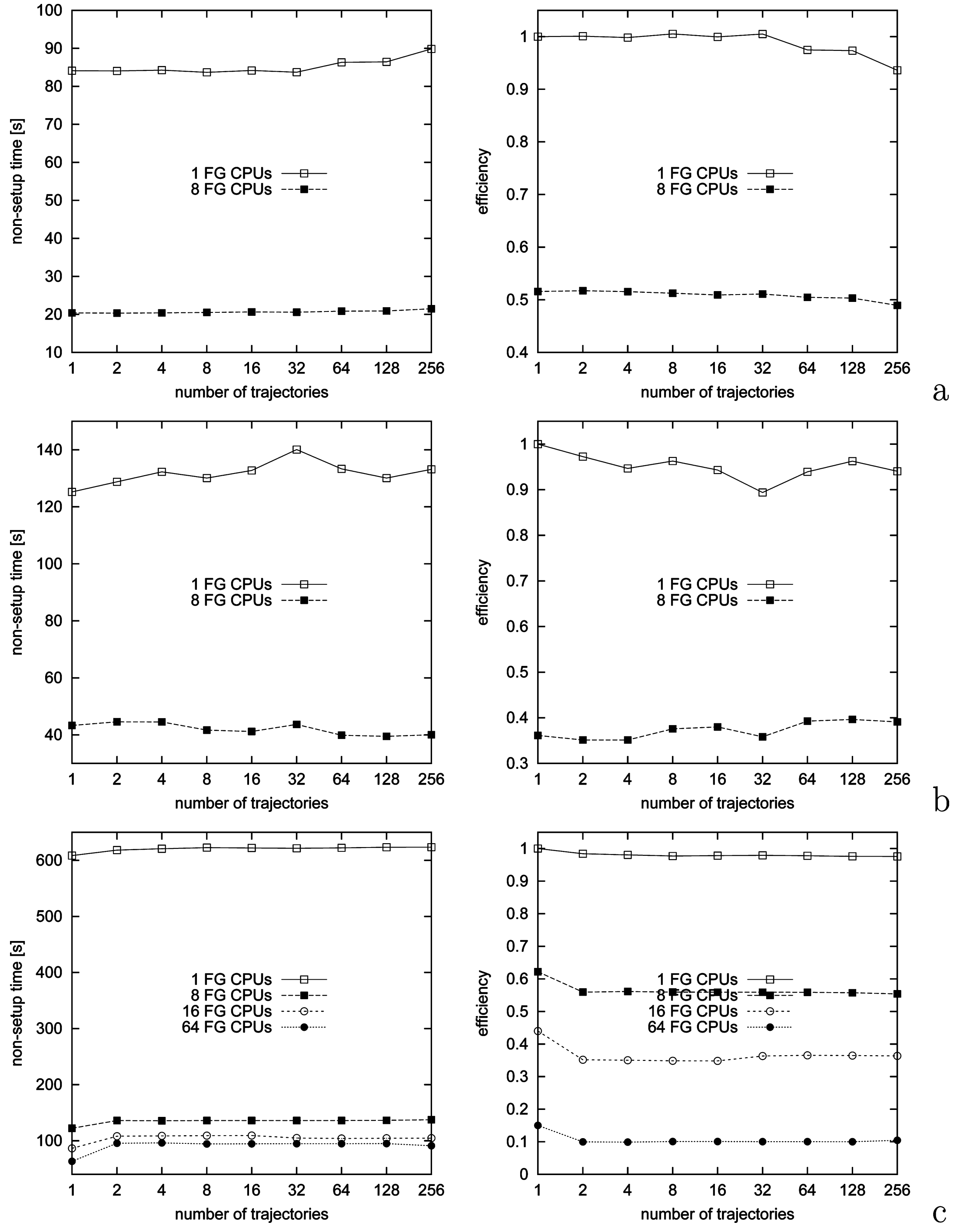


Figure 11. Plots of nonsetup times and efficiencies in 40 000-step MREMD simulations with replica exchange each 10 000 steps, numbers of trajectories from 1 (reference) to 256, and various numbers of fine-grain tasks per trajectory obtained for the 1KOY (62-residue) protein with (a) galera.task.gda.pl, (b) bigben.psc.edu, and (c) jugene.fz-juelich.de.

the drop of efficiency or the increase of the nonsetup time for IBM BlueGene/P is nearly constant. By detailed timing of the runs, we found that the decrease of efficiency is caused by increasing waiting times by the master processors which distributes coarse-grained tasks (Figure 2). Most probably, this waiting time increases because the calculations for each trajectory take slightly different times, which results in a load imbalance of the coarse-grained tasks.

4. Conclusions

In this work, we have extended the parallelization of UNRES MD from the already-existing one processor per trajectory mode,[51] by fine-graining single-trajectory calculations. Even for the smallest protein (1E0L, 37 residues), a speedup from about 2 to about 5 can be achieved depending on the machine (the largest on IBM BlueGene/P which has the fastest communication); for the largest protein studied (1TF5, 767

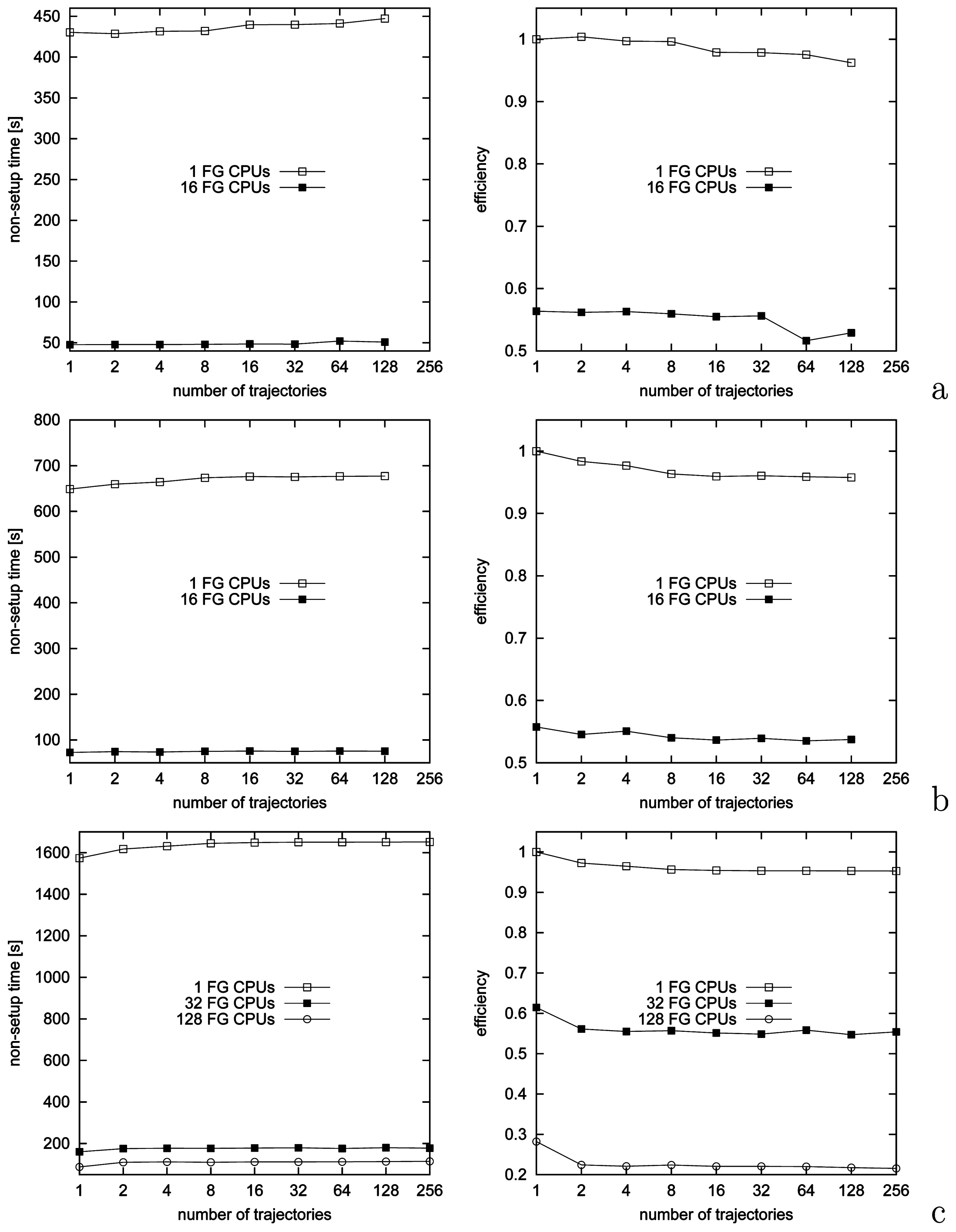


Figure 12. Plots of nonsetup times and efficiencies in 20 000-step MREMD simulations with replica exchange each 5000 steps, numbers of trajectories from 1 (reference) to 256, and various numbers of fine-grain tasks per trajectory obtained for the 1P1D (196-residue) protein with (a) galera.task.gda.pl, (b) bigben.psc.edu, and (c) jugene.fz-juelich.de.

residues), a 160-fold maximum speedup has been reached with 1024 processors/trajectory, and 120-fold speedup at 50% parallel efficiency with 256 processors/trajectory was reached with IBM BlueGene/P. Even for machines with slower communications (but 4- or 6-times greater processor speed than that of IBM BlueGene/P), a reasonable speedup of 32 or 16 can be achieved at 50% efficiency with the Cray XT3 and Xeon cluster, respectively. The efficiency is slightly diminished for MREMD runs which involve communication between coarse-grained tasks; however, this does not seem to be a significant issue. Given the fine-grained parallelism, the wall-clock time necessary to compute a single trajectory increases linearly with the number of residues, as opposed to an increase with the square of the number of residues when using a single processor per trajectory. For the 1TF5 protein, a single MD step can be accomplished in 0.015 wall-clock s, which means that a 20 000 000 step run (0.1 μs with 5 fs MD time step) can be accomplished in about 3.5 days of simulations. Given the fact that the UNRES (and, generally, a coarse-grained approach) time scale is about 1000 times wider than that of the experimental time scale (i.e., protein folding with UNRES takes only nanoseconds,[42,43] while the

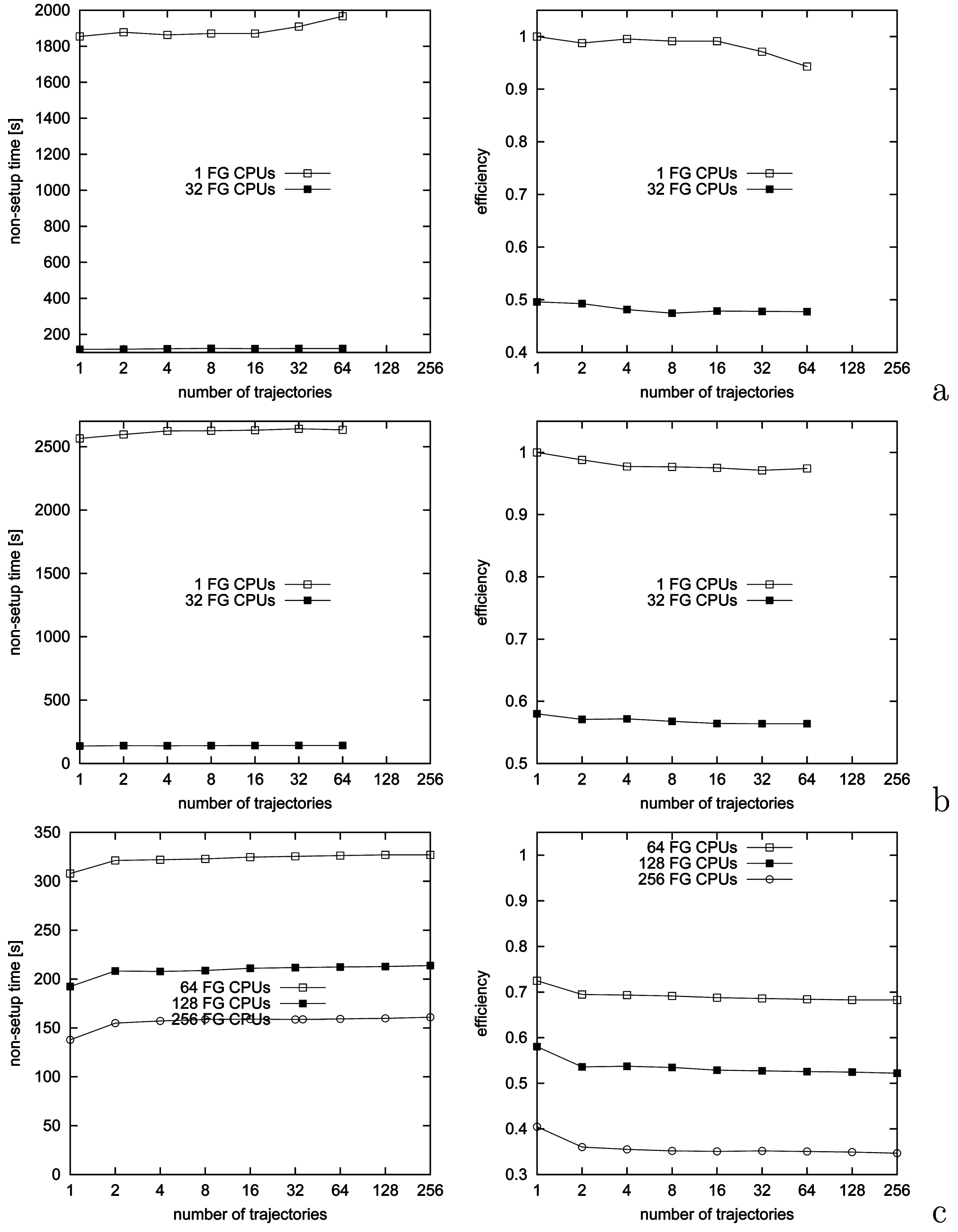


Figure 13. Plots of nonsetup times and efficiencies in 10 000-step MREMD simulations with replica exchange each 5000 steps, numbers of trajectories from 1 (reference) to 256, and various numbers of fine-grain tasks per trajectory obtained for the 2KHO (600-residue) protein with (a) galera.task.gda.pl, (b) bigben.psc.edu, and (c) jugene.fz-juelich.de. The reference time (1 trajectory with 1 fine-grained processor) corresponding to the runs of part c was calculated from the 2KHO entry in the data plotted in Figure 6.

fastest-folding proteins fold in microseconds),[44] this achievement enables us to carry out millisecond-scale simulations of large-size proteins in real time. Introducing a cutoff on nonbonded interactions, as pointed out in section 3.1, might push the achievable time scale even farther or, at least, reduce the cost of calculation, and we are currently working on this modification. As mentioned in section 3.1, the load balance can be achieved with a cutoff in the force-decomposition scheme;[63,64] however, we will also explore the domain-decomposition scheme.

At present, UNRES requires about 1 GB/processor memory to run calculations for ~900-residue proteins. A large part of this memory is occupied by matrix **G**, its inverse, and auxiliary matrices. This poses some problems, although a great part of this could be eliminated by switching to single precision (which, additionally, should reduce the CPU time) and by putting the arrays in shared memory. Because the array **G** for multichain proteins consists of independent blocks, each pertaining to a different chain, and the maximum number of residues per chain usually does not exceed 2000, this modification will solve a large part of the memory problem. The second substantial part of memory is required to store the information to compute the $U_{\mathrm{corr}}^{(4)}$ terms; however this part can be distributed between fine-grained processors. This work is presently being carried out in our laboratory.

The successful fine-grain parallel implementation of UNRES reported in this work suggests that the UNRES code can be ported efficiently to the GPUs, reaching an even greater speedup of the fine-grained part of the code than on IBM BlueGene/P; for all-atom MD, a 700-fold speedup was recently reported.[18] In particular, large constant arrays belonging to given conformations could be shared by all processors of a GPU unit, which would eliminate the present problem occurring with distributed-memory fine-grained UNRES. However, even though implementation of UNRES on the GPUs seems to be very attractive, it is not clear if a similar speedup can be achieved with UNRES as that with all-atom-code implementation on GPUs. We are currently in the process of porting UNRES code to GPUs.

Acknowledgment. This work was supported by grants from the Polish Ministry of Science and Higher Education (N N204 049035), the National Institutes of Health (GM-14312), and the National Science Foundation (MCB05-41633). This research was supported in part by the National Science Foundation through TeraGrid resources provided by Pittsburgh Supercomputing Center. Computational resources were also provided by (a) Argonne Leadership Computing Facility at Argonne National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under contract DE-AC02-06CH11357, (b) the John von Neumann Institute for Computing at the Central Institute for Applied Mathematics, Forschungszentrum Jülich, Germany, and (c) the Informatics Center of the Metropolitan Academic Network (IC MAN) in Gdańsk. The following local resources—(a) the Beowulf cluster at the Department of Computer Science, Cornell University, (b) our 800-processor Beowulf cluster at Baker Laboratory of Chemistry, Cornell University, and (c) our 45-processor Beowulf cluster at the Faculty of Chemistry, University of Gdańsk—were used to develop and test the code.

References

(1) McCammon, J. A.; Gelin, B. R.; Karplus, M. Dynamics of folded proteins. *Nature* **1977**, *267*, 585–590.

(2) Tirado-Rives, J.; Jorgensen, W. L. Molecular dynamics simulations of the unfolding of an α-helical analogue of ribonuclease A S-peptide in water. *Biochemistry* **1991**, *30*, 3864–3871.

(3) Daggett, V.; Levitt, M. Molecular dynamics simulations of helix denaturation. *J. Mol. Biol.* **1992**, *223*, 1121–1138.

(4) Brooks III, C. L. Characterization of native apomyoglobin by molecular dynamics simulations. *J. Mol. Biol.* **1992**, *227*, 375–380.

(5) Mark, A. E.; van Gunsteren, W. F. Simulation of the thermal-denaturation of hen egg-white lysozyme: trapping the molten globule state. *Biochemistry* **1992**, *31*, 7745–7748.

(6) Tirado-Rives, J.; Jorgensen, W. L. Molecular dynamics simulations of the unfolding of apomyoglobin in water. *Biochemistry* **1993**, *32*, 4175–4184.

(7) Daggett, V.; Levitt, M. Protein unfolding pathways explored through molecular-dynamics simulations. *J. Mol. Biol.* **1993**, *232*, 600–619.

(8) Day, R.; Daggett, V. All-atom simulations of protein folding and unfolding. *Adv. Protein Chem.* **2003**, *66*, 373–403.

(9) Snow, C. D.; Sorin, E. J.; Rhee, Y. M.; Pande, V. S. How well can simulation predict protein folding kinetics and thermodynamics. *Annu. Rev. Biophys. Biomol. Struct.* **2005**, *34*, 43–69.

(10) Adcock, S. A.; McCammon, J. A. Molecular dynamics: survey of methods for simulating the activity of proteins. *Chem. Rev.* **2006**, *106*, 1589–1615.

(11) Daggett, V. Protein folding simulations. *Chem. Rev.* **2006**, *106*, 1898–1916.

(12) Scheraga, H. A.; Khalili, M.; Liwo, A. Protein-folding dynamics: Overview of molecular simulation techniques. *Annu. Rev. Phys. Chem.* **2007**, *58*, 57–83.

(13) Kmiecik, S.; Kolinski, A. Folding pathway of the B1 domain of protein G explored by multiscale modeling, *Biophys. J.* **2008**, *94*, 726–736.

(14) Pande, V. S.; Baker, I.; Chapman, J.; Elmer, S.; Kaliq, S.; Larson, S. M.; Rhee, Y. M.; Shirts, M. R.; Snow, C. D.; Sorin, E. J.; Zagrovic, B. Atomistic protein folding simulations on the submillisecond timescale using worldwide distributed computing. *Biopolymers* **2003**, *68*, 91–109.

(15) Hess, B.; Kutzner, C.; van der Spoel, D.; Lindahl, E. GROMACS 4: Algorithms for highly efficient, load-balanced, and scalable molecular simulation. *J. Chem. Theory Comput.* **2008**, *4*, 435–447.

(16) Phillips, J. C.; Braun, R.; Wang, W.; Gumbart, J.; Tajkhorshid, E.; Villa, E.; Chipot, C.; Skeel, R. D.; Kalé, L.; Schulten, K. Scalable molecular dynamics with NAMD. *J. Comput. Chem.* **2005**, *26*, 1781–1802.

(17) Bowers, K. J.; Chow, E.; Xu, H.; Dror, R. O.; Eastwood, M. P.; Gregersen, B. A.; Klepeis, J. L.; Kolossvary, I.; Moraes, M. A.; Sacerdoti, F. D.; Salmon, J. K.; Shan, Y.; Shaw, D. E. Scalable algorithms for molecular dynamics simulations on commodity clusters. *ACM/IEEE SC 2006 Conference (SC.06)* **2006**, 43.

(18) Friedrichs, M. S.; Eastman, P.; Vaidyanathan, V.; Houston, M.; Legrand, S.; Beberg, A. L.; Ensign, D. L.; Bruns, C. M.; Pande, V. S. Accelerating molecular dynamic simulation on graphics processing units. *J. Comput. Chem.* **2009**, *30*, 864–872.

(19) Shaw, D. E.; Deneroff, M. M.; Dror, R. O.; Kuskin, J. S.; Larson, R. H.; Salmon, J. K.; Young, C.; Batson, B.; Bowers, K. J.; Chao, J. C.; Eastwood, M. P.; Gagliardo, J.; Grossman, J. P.; Ho, C. R.; Ierardi, D. J.; Kolossvary, I.; Klepeis, J. L.; Layman, T.; Mcleavey, C.; Moraes, M. A.; Mueller, R.; Priest, E. C.; Shan, Y.; Spengler, J.; Theobald, M.; Towles, B.; Wang, S. C. Anton, a special-purpose machine for molecular dynamics simulation. *Commun. ACM* **2008**, *51*, 91–97.

(20) Klepeis, J. L.; Lindorff-Larsen, K.; Dror, R. O.; Shaw, D. E. Long-timescale molecular dynamics simulations of protein structure and function. *Curr. Opin. Struct. Biol.* **2009**, *19*, 120–127.

(21) Kolinski, A.; Skolnick, J. Reduced models of proteins and their applications. *Polymer* **2004**, *45*, 511–524.

(22) Tozzini, V. Coarse-grained models for proteins. *Curr. Opin. Struct. Biol.* **2005**, *15*, 144–150.

(23) Colombo, G.; Micheletti, C. Protein folding simulations: combining coarse-grained models and all-atom molecular dynamics. *Theor. Chem. Acc.* **2006**, *116*, 75–86.

(24) Ayton, G. S.; Noid, W. G.; Voth, G. A. Multiscale modeling of biomolecular systems: in serial and in parallel. *Curr. Opin. Struct. Biol.* **2007**, *17*, 192–198.

(25) Clementi, C. Coarse-grained models of protein folding: toy models or predictive tools. *Curr. Opin. Struct. Biol.* **2008**, *18*, 10–15.

(26) Voth, G. Introduction. In *Coarse-Graining of Condensed Phase and Biomolecular Systems*, 1st ed.; Voth, G., Ed; CRC Press, Taylor & Francis: Boca Raton, FL, 2008; pp 1−4.

(27) Liwo, A.; Ołdziej, S.; Pincus, M. R.; Wawak, R. J.; Rackovsky, S.; Scheraga, H. A. A united-residue force field for off-lattice protein-structure simulations. I. Functional forms and parameters of long-range side-chain interaction potentials from protein crystal data. *J. Comput. Chem.* **1997**, *18*, 849–873.

(28) Liwo, A.; Pincus, M. R.; Wawak, R. J.; Rackovsky, S.; Ołdziej, S.; Scheraga, H. A. A united-residue force field for off-lattice protein-structure simulations. II: Parameterization of local interactions and determination of the weights of energy terms by Z-score optimization. *J. Comput. Chem.* **1997**, *18*, 874–887.

(29) Liwo, A.; Kaźmierkiewicz, R.; Czaplewski, C.; Groth, M.; Ołdziej, S.; Wawak, R. J.; Rackovsky, S.; Pincus, M. R.; Scheraga, H. A. United-residue force field for off-lattice protein-structure simulations; III. Origin of backbone hydrogen-bonding cooperativity in united-residue potentials. *J. Comput. Chem.* **1998**, *19*, 259–276.

(30) Liwo, A.; Czaplewski, C.; Pillardy, J.; Scheraga, H. A. Cumulant-based expressions for the multibody terms for the correlation between local and electrostatic interactions in the united-residue force field. *J. Chem. Phys.* **2001**, *115*, 2323–2347.

(31) Liwo, A.; Arłukowicz, P.; Czaplewski, C.; Ołdziej, S.; Pillardy, J.; Scheraga, H. A. A method for optimizing potential-energy functions by a hierarchical design of the potential-energy landscape: Application to the UNRES force field. *Proc. Natl. Acad. Sci. U.S.A.* **2002**, *99*, 1937–1942.

(32) Ołdziej, S.; Kozłowska, U.; Liwo, A.; Scheraga, H. A. Determination of the potentials of mean force for rotation about $\mathbf{C}^{\alpha}\cdots\mathbf{C}^{\alpha}$ virtual bonds in polypeptides from the *ab initio* energy surfaces of terminally-blocked glycine, alanine, and proline. *J. Phys. Chem. A* **2003**, *107*, 8035–8046.

(33) Liwo, A.; Ołdziej, S.; Czaplewski, C.; Kozłowska, U.; Scheraga, H. A. Parameterization of backbone-electrostatic and multibody contributions to the UNRES force field for protein-structure prediction from *ab initio* energy surfaces of model systems. *J. Phys. Chem. B* **2004**, *108*, 9421–9438.

(34) Ołdziej, S.; Łagiewka, J.; Liwo, A.; Czaplewski, C.; Chinchio, M.; Nanias, M.; Scheraga, H. A. Optimization of the UNRES force field by hierarchical design of the potential-energy landscape. 3. Use of many proteins in optimization. *J. Phys. Chem. B* **2004**, *108*, 16950–16959.

(35) Kozłowska, U.; Liwo, A.; Scheraga, H. A. Determination of virtual-bond-angle potentials of mean force for coarse-grained simulations of protein structure and folding from ab initio energy surfaces of terminally-blocked glycine, alanine, and proline. *J. Phys.: Condens. Matter* **2007**, *19*, 285203.

(36) Liwo, A.; Khalili, M.; Czaplewski, C.; Kalinowski, S.; Ołdziej, S.; Wachucik, K.; Scheraga, H. A. Modification and optimization of the united-residue (UNRES) potential energy function for canonical simulations. I. Temperature dependence of the effective energy function and tests of the optimization method with single training proteins. *J. Phys. Chem. B* **2007**, *111*, 260–285.

(37) Liwo, A.; Czaplewski, C.; Ołdziej, S.; Rojas, A. V.; Kaḿierkiewicz, R.; Makowski, M.; Murarka, R. K.; Scheraga, H. A. Simulation of protein structure and dynamics with the coarse-grained UNRES force field. In *Coarse-Graining of Condensed Phase and Biomolecular Systems*, 1st ed.; Voth, G., Ed; CRC Press, Taylor & Francis: Boca Raton, FL, 2008; pp 1391−1411.

(38) Kozłowska, U.; Liwo, A.; Scheraga, H. A. Determination of side-chain-rotamer and virtual-bond-stretching potentials of mean force for coarse-grained simulations of protein structure and folding from AM1 energy surfaces of terminally-blocked amino-acid residues. 1. The Method. *J. Comput. Chem.* **2010**, in press; DOI: 10.1002/jcc.21399.

(39) Kozłowska, U.; Maisuradze, G. G.; Liwo, A.; Scheraga, H. A. Determination of side-chain-rotamer and virtual-bond-stretching potentials of mean force for coarse-grained simulations of protein structure and folding from AM1 energy surfaces of terminally-blocked amino-acid residues. 2. Results, comparison with statistical potentials, and implementation, *J. Comput. Chem.* **2010**, in press, DOI: 10.1002/jcc.21402.

(40) Ołdziej, S.; Czaplewski, C.; Liwo, A.; Chinchio, M.; Nanias, M.; Vila, J. A.; Khalili, M.; Arnautova, Y. A.; Jagielska, A.; Makowski, M.; Schafroth, H. D.; Kaḿierkiewicz, R.; Ripoll, D. R.; Pillardy, J.; Saunders, J. A.; Kang, Y. K.; Gibson, K. D.; Scheraga, H. A. Physics-based protein-structure prediction using a hierarchical protocol based on the UNRES force field: Assessment in two blind tests. *Proc. Natl. Acad. Sci. U.S.A.* **2005**, *102*, 7547–7552.

(41) Khalili, M.; Liwo, A.; Rakowski, F.; Grochowski, P.; Scheraga, H. A. Molecular dynamics with the united-residue model of polypeptide chains. I. Lagrange equations of motion and tests of numerical stability in the microcanonical mode. *J. Phys. Chem. B* **2005**, *109*, 13785–13797.

(42) Khalili, M.; Liwo, A.; Jagielska, A.; Scheraga, H. A. Molecular dynamics with the united-residue model of polypeptide chains. II. Langevin and Berendsen-bath dynamics and tests on model α-helical systems. *J. Phys. Chem. B* **2005**, *109*, 13798–13810.

(43) Liwo, A.; Khalili, M.; Scheraga, H. A. Ab initio simulations of protein-folding pathways by molecular dynamics with the united-residue model of polypeptide chains. *Proc. Natl. Acad. Sci. U.S.A.* **2005**, *102*, 2362–2367.

(44) Kubelka, J.; Hofrichter, J.; Eaton, W. A. The protein folding 'speed limit'. *Curr. Opinion Struct. Biol.* **2004**, *14*, 76–88.

(45) Khalili, M.; Liwo, A.; Scheraga, H. A. Kinetic studies of folding of the B-domain of staphylococcal protein A with molecular dynamics and a united-residue (UNRES) model of polypeptide chains. *J. Mol. Biol.* **2006**, *355*, 536–547.

(46) Cieplak, M.; Hoang, T. X.; Robbins, M. O. Thermal folding and mechanical unfolding pathways of protein secondary structures. *Proteins: Struct., Funct., Genet.* **2002**, *49*, 104–113.

(47) Brown, S.; Fawzi, N. J.; Head-Gordon, T. Coarse-grained sequences for protein folding and design. *Proc. Natl. Acad. Sci. U.S.A.* **2003**, *100*, 10712–10717.

(48) Hansmann, U. H. E.; Okamoto, Y. Comparative study of multicanonical and simulated annealing algorithms in the protein folding problem. *Physica A* **1994**, *212*, 415–437.

(49) Rhee, Y. M.; Pande, V. S. Multiplexed-replica exchange molecular dynamics method for protein folding simulation. *Biophys. J.* **2003**, *84*, 775–786.

(50) Nanias, M.; Czaplewski, C.; Scheraga, H. A. Replica Exchange and Multicanonical Algorithms with the Coarse-Grained United-Residue (UNRES) Force Field. *J. Chem. Theory Comput.* **2006**, *2*, 513–528.

(51) Czaplewski, C.; Kalinowski, S.; Liwo, A.; Scheraga, H. A. Application of multiplexing replica exchange molecular dynamics method to the UNRES force field: tests with α and $\alpha + \beta$ proteins. *J. Chem. Theory Comput.* **2009**, *5*, 627–640.

(52) Nishikawa, K.; Momany, F. A.; Scheraga, H. A. Low-energy structures of two dipeptides and their relationship to bend conformations. *Macromolecules* **1974**, *7*, 797–806.

(53) Kubo, R. Generalized cumulant expansion method. *J. Phys. Soc. Jpn.* **1962**, *17*, 1100–1120.

(54) Kolinski, A.; Skolnick, J. Discretized model of proteins. I. Monte Carlo study of cooperativity in homopolypeptides. *J. Chem. Phys.* **1992**, *97*, 9412–9426.

(55) Shen, H.; Liwo, A.; Scheraga, H. A. An improved functional form for the temperature scaling factors of the components of the mesoscopic UNRES force field for simulations of protein structure and dynamics. *J. Phys. Chem. B* **2009**, *113*, 8738–8744.

(56) He, Y.; Xiao, Y.; Liwo, A.; Scheraga, H. A. Exploring the parameter space of the coarse-grained UNRES force field by random search: Selecting a transferable medium-resolution force field. *J. Comput. Chem.* **2009**, *30*, 2127–2135.

(57) Berendsen, H. J. C.; Postma, J. P. M.; van Gunsteren, W. F.; DiNola, A.; Haak, J. R. Molecular dynamics with coupling to an external bath. *J. Chem. Phys.* **1984**, *81*, 3684–3690.

(58) Lee, J.; Scheraga, H. A. Conformational space annealing by parallel computations: extensive conformational search of Met-enkephalin and of the 20-residue membrane-bound portion of melittin. *Int. J. Quantum Chem.* **1999**, *75*, 255–265.

(59) Czaplewski, C.; Liwo, A.; Pillardy, J.; Ołdziej, S.; Scheraga, H. A. Improved Conformational Space Annealing method to treat β-structure with the UNRES force-field and to enhance scalability of parallel implementation. *Polymer* **2004**, *45*, 677–686.

(60) Gropp, W.; Lusk, E.; Skjellum, A. Parallel libraries. In *Using MPI. Portable Parallel Programming with the Message-Passing Interface*, 2nd edition; The MIT Press; Cambridge, MA, 1999; pp 157−193.

(61) Blackford, L. S.; Demmel, J.; Dongarra, J.; Duff, I.; Hammarling, S.; Henry, G.; Heroux, M.; Kaufman, L.; Lumsdaine, A.; Petitet, A.; Pozo, R.; Remington, K.; Whaley, R. C. An updated set of basic linear algebra subprograms (BLAS). *ACM Trans. Math. Soft.* **2002**, *28−2*, 135–151.

(62) Rakowski, F.; Grochowski, P.; Lesyng, B.; Liwo, A.; Scheraga, H. A. Implementation of a symplectic multiple-time-step molecular dynamics algorithm, based on the united-residue mesoscopic potential energy function. *J. Chem. Phys.* **2006**, *125*, 204107.

(63) Plimpton, S. Fast parallel algorithms for short-range molecular dynamics. *J. Comput. Phys.* **1995**, *117*, 1–19.

(64) Plimpton, S.; Hendrickson, B. A new parallel method for molecular dynamics simulation of macromolecular systems. *J. Comput. Chem.* **1996**, *17*, 326–337.

(65) Hendrickson, B.; Leland, R. An improved spectral graph partitioning algorithm for mapping parallel computations. *SIAM J. Sci. Comput.* **1995**, *16*, 452–469.

(66) Amdahl, G. The validity of the single processor approach to achieving large-scale computing capabilities. In *Proceedings of AFIPS Spring Joint Computer Conference*, Atlantic City, NJ; AFIPS Press: 1967; pp 483−485.

CT9004068