Evaluating Polarizable Potentials on Distributed Memory Parallel Computers: Program Development and Applications

DAN N. BERNARDO, YANBO DING, KARSTEN KROGH-JESPERSEN,* and RONALD M. LEVY*

Department of Chemistry, Rutgers, The State University of New Jersey, New Brunswick, New Jersey 08903

Received 29 September 1994; accepted 19 December 1994

ABSTRACT

The efficient evaluation of polarizable molecular mechanics potentials on distributed memory parallel computers is discussed. The program executes at 7–10 Mflops/node on a 32-node CM-5 partition and is 19 times faster than comparable code running on a single-processor HP 9000/735. On the parallel computer, matrix inversion becomes a practical alternative to the commonly used iterative method for the calculation of induced dipole moments. The former method is useful in cases such as free-energy perturbation (FEP) simulations, which require highly accurate induced dipole moments. Matrix inversion is performed 110 times faster on the CM-5 than on the HP. We show that the accuracy which is needed for FEP calculations with polarization can be obtained by either matrix inversion or by performing a large number of iteration cycles to satisfy convergence tolerances that are less than 10^{-6} D.

Introduction

The use of many-body, polarizable potentials in molecular mechanics calculations is slowly increasing.[1–11] The attractive feature of such potentials lies in their ability to model electronic contributions to the redistribution of the charge density within a molecule in response to changes in the molecular environments during a simulation. A major drawback to the use of such potentials is the amount of computational power required to evaluate the induced dipole moments as well as the energies and forces due to electronic polarization. Typical algorithms for determining the induced dipole moments involve either iteration or matrix inversion. For a system with N polarizable sites, the number of operations for these methods scales as N^2 and N^3, respectively, and the runtimes required to apply polarizable potentials to large systems can become prohibitively long. There is also a significant increase in memory requirements associated with the use of polarizable potentials. Most of the methods for calcu-

*Author to whom all correspondence should be addressed.

CCC 0192-8651 / 95 / 091141-12

lating polarization energies and forces require construction of the dipole tensor and derivative matrices, the sizes of which scale as N^2. As a result, current studies have been limited to simulations involving small systems and/or short durations.

Limitations due to the twin bottlenecks of computational speed and memory usage can be circumvented to some extent by using parallel computers. It is obvious that the computational loads can be distributed among many processors; however, on distributed memory machines the overall efficiency depends on minimizing interprocessor communication. The allocation of memory as well as computational requirements among numerous processors is not always straightforward; suboptimal program design can lead to wasted processor time as well as excessive use of machine memory. In this article, we discuss how the terms used in setting up the calculations for induced dipole moments (as well as the two-body Coulomb and Lennard-Jones interactions) can be distributed among processors in a pattern which speeds up memory access and minimizes interprocessor communications in the later stages of the calculation.

The use of parallel processing in molecular mechanics simulations is not a new concept; numerous studies on this topic have been published.[12-22] For example, Mertz et al. distributed the computational load over several processors.[12] A very different approach was used by DeBolt and co-workers, who performed independent calculations on each processor.[13] Other studies have concentrated on the parallel computation of neighbor lists and energies for nonbonded pairwise interactions. In the replicated-data method,[14-18] each processor receives a copy of all atomic coordinates. The spatial-decomposition method[19-21] divides the simulation domain into pieces and assigns each to a processor. Each processor computes the energies and forces of atoms in its own domain. Finally, the force-decomposition method uses block operations on the force matrix which reduce memory and communication costs.[22] The aforementioned cases involved the use of parallel processing machines for faster evaluation of existing molecular mechanics force fields. This article concentrates on a different problem: We use parallel processing to compute energies and forces associated with the use of an all-atom, many-body polarizable potential, the evaluation of which, for large chemical systems, is either extremely compute intensive (when the iterative approach is used) or impractical (when matrix inversion is used) on the current generation of serial machines. An article dealing with the data parallel implementation of a polarizable potential has appeared.[23]

This article addresses some of the computational aspects associated with the use of polarizable potentials in molecular mechanics simulations and demonstrates how these potentials can be efficiently evaluated on a distributed memory parallel computer. Comparisons to simulations on serial machines are made, and some differences in program development for serial and parallel machines are discussed. It is shown that long simulations involving polarizable potentials can be performed on a routine basis if parallel processing is used to bypass the important computational bottlenecks. Finally, the relative merits of using the iterative and matrix inversion approaches to the calculation of induced dipoles are discussed.

Theory

In this section we briefly outline the procedure for calculating the potential energy of a polarizable molecular system. A more complete description of the polarizable potential as well as values of the various force field parameters can be found in refs. 1, 24, and 25. The total potential energy of the system is written as

$$U_{\text{tot}} = U_{\text{pair}} + U_{\text{pol}} \tag{1}$$

The pairwise term describes the Lennard-Jones and Coulomb interactions

$$U_{\text{pair}} = \sum_{i,j} 4\epsilon_{ij}\left[\left(\frac{\sigma_{ij}}{r_{ij}}\right)^{12} - \left(\frac{\sigma_{ij}}{r_{ij}}\right)^{6}\right] + \frac{q_i q_j}{r_{ij}} \tag{2}$$

where r_{ij} is the distance between atoms i and j, q_i and q_j are partial atomic charges, and ϵ_{ij} and σ_{ij} are the Lennard-Jones well depths and radii, respectively, for this particular i–j pair.

The polarization energy U_{pol} is expressed as

$$U_{\text{pol}} = -\frac{1}{2}\sum_i \mathbf{u}_i \cdot \mathbf{E}_i^0 \tag{3}$$

Given the electrostatic field due to the permanent charges $\mathbf{E}_i^0$, the induced dipole moments $\boldsymbol{\mu}_i$ can be obtained by procedures based on the atom–dipole interaction model. If the polarizability tensor of

atom i is α_i, then the induced dipole moment $\boldsymbol{\mu}_i$ is given by

$$\boldsymbol{\mu}_i = \boldsymbol{\alpha}_i\left(\mathbf{E}_i^0 - \sum_{j\neq i}\mathbf{T}_{ij}\boldsymbol{\mu}_j\right) \tag{4}$$

The dipole field tensor $\mathbf{T}_{ij}$ is calculated using a modified version of Thole's procedure[1,24]

$$\mathbf{T}_{ij} = \frac{4w_{ij}^3 - 3w_{ij}^4}{r_{ij}^3}\mathbf{I} - \frac{3w_{ij}^4}{r_{ij}^5}\begin{bmatrix} x^2 & xy & xz \\ xy & y^2 & yz \\ xz & yz & z^2 \end{bmatrix} \tag{5}$$

where $\mathbf{I}$ is the identity matrix and x, y, and z are the Cartesian components of r_{ij}. The scaling function w_{ij} is

$$w_{ij} = \begin{cases} \dfrac{r_{ij}}{s_{ij}} + \dfrac{1}{m}\left[1 - \left(\dfrac{r_{ij}}{s_{ij}}\right)^m\right] & \text{if } r_{ij} < s_{ij} \\ 1 & r_{ij} \geq s_{ij} \end{cases} \tag{6}$$

which uses a scaling distance based on the isotropic polarizabilities (α)

$$s_{ij} = 1.662(\alpha_i\alpha_j)^{1/6} \tag{7}$$

The most frequently used algorithm for the calculation of $\boldsymbol{\mu}$ involves the iterative solution of eq. (4). Its application is a specific case of Jacobi's method for obtaining solutions to linear equations via relaxation, a method which is known to converge slowly.[26] We find that for a large number of polarizable sites ($N > 20$), use of eq. (4) leads to iterated values for $\boldsymbol{\mu}$ which oscillate about their correct values. To reduce this oscillatory behavior, we institute a simple form of damping. Improved values of $\boldsymbol{\mu}$ in the mth iteration are obtained by

$$\boldsymbol{\mu}'_m = (1 - b)\boldsymbol{\mu}_m + b\boldsymbol{\mu}_{m-1} \tag{8}$$

where b dictates the extent of damping and is given by the empirical set of equations

$$b = \begin{cases} 0 & \text{if } m < 3 \\ (m-3)/24 & \text{if } 3 \leq m \leq 15 \\ 1/2 & \text{if } m > 15 \end{cases} \tag{9}$$

This stabilizes the results of the iterative process and allows $\boldsymbol{\mu}$ to converge to the correct values. For determining the convergence of results, we evaluate the root mean square difference between induced dipole moments resulting from successive iterations and require that this value be less than some convergence criterion

$$\delta \geq \left\langle (\mu_m - \mu_{m-1})^2 \right\rangle^{1/2} \tag{10}$$

Caldwell et al.[3] suggest using $\delta = 0.01$ D/atom. With this value of δ, an average of about eight iterations are necessary to attain convergence to this value in a system containing 216 water molecules. One can, of course, devise more elaborate and possibly faster procedures for performing the iteration, an issue which we are exploring.

The iterative approach does not always yield the correct values of the induced dipole moments. Failure of the procedure is usually due to close encounters between two polarizable sites, which result in unusually large induced dipole moments. In contrast, the matrix inversion procedure is subject only to computational constraints and can always be used to obtain the correct values. We first rearrange eq. (4) and write it in matrix form as

$$\begin{bmatrix} \boldsymbol{\alpha}_1^{-1} & \mathbf{T}_{12} & \cdots & \mathbf{T}_{1N} \\ \mathbf{T}_{21} & \boldsymbol{\alpha}_2^{-1} & \cdots & \mathbf{T}_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{T}_{N1} & \mathbf{T}_{N2} & \cdots & \boldsymbol{\alpha}_N^{-1} \end{bmatrix}\begin{bmatrix} \boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_2 \\ \vdots \\ \boldsymbol{\mu}_N \end{bmatrix} = \begin{bmatrix} \mathbf{E}_1^0 \\ \mathbf{E}_2^0 \\ \vdots \\ \mathbf{E}_N^0 \end{bmatrix} \tag{11}$$

or briefly

$$\mathbf{A}\boldsymbol{\mu} = \mathbf{E}^0 \tag{12}$$

Once $\mathbf{A}$ is constructed, $\boldsymbol{\mu}$ can be obtained by matrix inversion

$$\boldsymbol{\mu} = \mathbf{A}^{-1}\mathbf{E}^0 \tag{13}$$

The matrix inversion approach is more compute intensive than the iterative approach. Note that $\mathbf{A}$ is a $3N \times 3N$ matrix (where N is the number of polarizable sites). As mentioned earlier, the number of floating point operations required in the iterative approach scales as N^2, whereas the matrix inversion approach scales as N^3.

The calculation of the induced dipole moments is not the only time-consuming step which arises in molecular simulations with polarizable potentials. Construction of the dipole tensor matrix $\mathbf{A}$, although done only once every molecular dynamics timestep, also takes a significant amount of computer time since every interatomic distance

must be calculated and corrected for periodic boundary conditions when appropriate. Finally, a calculation of the forces

$$\mathbf{F}_k = \sum_i \boldsymbol{\mu}_i(\nabla_k \mathbf{E}_i^0) - \frac{1}{2}\sum_i \left[\boldsymbol{\mu}_i \sum_j (\nabla_k \mathbf{T}_{ij})\boldsymbol{\mu}_j\right] \quad (14)$$

requires evaluation of the gradient of **A**.

The memory required to store **A** and its derivatives is large, particularly for models such as ours which assign polarizable sites to all the atomic centers. For a system of 216 water molecules, **A** becomes a 1944×1944 matrix which, in double precision, needs 30 Mb for storage. The same amount of memory is required for storing each of the three derivative matrices. One can avoid these memory problems by computing the elements of **A** and its derivatives only when their values are needed. However, this approach increases the computational requirements by a considerable amount since individual elements of the aforementioned matrices are needed several times. For example, **A** is used repeatedly when calculating the induced dipole moments via the iterative method.

The aforementioned polarization calculations represent only part of the calculations that are performed during a molecular dynamics simulation. Routines for calculating the energies and forces due to bond stretching and bending may be included, and procedures for maintaining the system temperature, neighbor lists, and I/O of atomic coordinates as well as other structural and thermodynamic information on the system are also required. For the results described later, these other tasks are performed with the IMPACT program.[27] The details of IMPACT will not be discussed in this article; we shall instead focus on the procedures used for implementing the demanding polarization calculations.

Results and Discussion

DIFFERENCES IN PROGRAM DEVELOPMENT FOR SERIAL AND PARALLEL MACHINES

The design and optimization of code to perform polarization calculations depends on the architecture of the computer. Most programmers are accustomed to working with serial machines. As a result, the task of writing code for computers of this architecture becomes relatively straightforward. For example, it is intuitive to define **A** as `a(3*N,3*N)` matrix to allow stride-1 addressing and possible vectorization. Some machines require proper nesting of `do` loops to allow vectorization for loop unrolling, but all these programming techniques are relatively well established.

Programming for parallel machines is more complex. Two commonly cited types of parallelism[28] are (1) control parallelism, in which two or more different operations can be performed simultaneously by separate sets of processors; and (2) data parallelism, in which the same operation is performed on many data elements by many processors. We have coded the polarizable potential using a data parallel language, CM Fortran,[29] for a number of reasons. First, the amount of data to be processed are far greater than the number of operations that can be performed concurrently. For example, the construction of **A** involves the operations represented by eqs. (5) and (6) on thousands of $\mathbf{r}_{ij}$ vectors. The data parallel model is well suited to handling this problem. Second, code for the data parallel model is similar to the code for serial machines and simplifies the development of programs for parallel machines.

Our parallel implementation of a polarizable molecular mechanics potential has been developed for the CM-5 computer made by Thinking Machines Corporation. A typical configuration for this computer involves the coordinated use of several processing nodes, each of which may have up to 32 Mb of memory and four vector units. When the vector units are present, each node is theoretically capable of performing 128 Mflops, although a realistic limit of 64.1 Mflops has been obtained.[30] The difference between the theoretical and actual ratings has been attributed to limits on memory access time and overhead due to switching between vector units. In any event, the amount of available computational power is considerable. For efficient use of the processors, this increase in computational speed must be accompanied by increases in communication speed. This is achieved by placing the nodes on a high-bandwidth network and by distributing memory (i.e., each node has its own memory). The processing nodes are supervised by a control processor, which handles program loading, control, and synchronization. The entire system is usually subdivided into sets of nodes,with each set having its own control processor. In this case, the set of nodes is called a "partition," and the control processor is known as the "partition manager." We have worked with partition sizes of 16, 32, 64, 128, and 256 nodes. Most of our production runs are made with 32- and 64-node partitions. IMPACT runs on the partition manager,

whereas the polarization terms as well as the Coulomb and Lennard-Jones interactions are calculated on the nodes.

The partition size affects the computational speed of an application since the number of nodes influences the amount of time spent in interprocessor communications. There are two interprocessor communications networks: (1) the Control Network, which handles operations that involve all the nodes at once, such as synchronization; and (2) the Data Network, which is used for data transfers in which each item may have a different source and destination. The maximum speed at which a node can transfer data to any other node via the Data Network is 14 Mb/s.[31] If one assumes that a floating point operation requires at least 16 bytes of data, then calculations which require interprocessor communication can proceed no faster than 0.88 Mflops. As mentioned earlier, each node is capable of performing 64.1 Mflops. In other words, the process of sending data from one node to another is roughly 2 orders of magnitude slower than the process of performing floating point operations involving data that already reside on the node. It is thus essential to minimize interprocessor communications in order to take advantage of the computational power present at each node.

The difference between computational and data transmission speeds leads to some nonintuitive programming decisions for the polarizable molecular mechanics calculations. For example, **A** is a symmetric matrix, and on serial machines it is more efficient to calculate half the elements and fill in the other half via symmetry. On the CM-5, this method would involve substantial amounts of general data movement, and it is therefore much more efficient to calculate all elements of **A** simultaneously using the data already present on each node.

The elements of **A** and its derivative matrices are divided among the nodes. The manner in which the elements are laid out is specified via software. This is important for the following reason: The large matrix **A** consists of numerous 3×3 submatrices $\mathbf{T}_{ij}$. Each $\mathbf{T}_{ij}$ is a function of $\mathbf{r}_{ij}$, α_i, and α_j (i.e., quantities which depend only on i and j). Once these quantities are loaded onto a node, the elements of $\mathbf{T}_{ij}$ and its derivatives can be calculated without any interprocessor communications if one defines **A** as `a(N,N,3,3)` array and uses compiler directives to ensure that the two rightmost dimensions are allocated within the memory of the same node. In other words, the nine elements `a(i,j,1,1),a(i,j,1,2),...,a(i,j,3,3)` correspond to the elements of $\mathbf{T}_{ij}$, and each submatrix can be constructed and used with no data movement between nodes.

The manner in which data are allocated among the nodes is extremely important. If **A** is defined as `a(3*N,3*N)` array (i.e., the same format used in programming for serial machines), it is no longer possible to use compiler directives to assign all the elements of each 3×3 submatrix to the same node. As a result, there is a substantial increase in the amount of interprocessor communication during the construction of **A** and its derivatives. We found that use of this inefficient format degrades the program performance by about a factor of 100 and increases memory usage by a factor of 2 to 3.

The methods used for the efficient calculation of the polarization energies and forces are also used to evaluate the pairwise components of the nonbonded energy (Coulomb and Lennard-Jones). These energies are derived from quantities such as $\mathbf{r}_{ij}$, which are evaluated during the construction of the dipole tensor matrix. Other parameters, such as the atomic charges and the Lennard-Jones well depths and radii, are loaded into arrays at the beginning of the simulation. The energies in eq. (2) are then evaluated via a sequence of matrix operations involving these parameters and $\mathbf{r}_{ij}$. These operations are efficiently executed on a parallel processing machine such as the CM-5 since each pairwise interaction is evaluated on a specific node.

COMPARISON OF BENCHMARKS

To examine differences between calculations performed in parallel and in a serial manner, we conducted benchmark runs on two serial computers (Silicon Graphics 4D/340 and Hewlett-Packard 9000/735) as well as 32- and 64-node partitions of a CM-5. A brief description of the four systems is included in Table I. The parallel computer has far more raw computational power and memory than the serial machines; in this section we shall discuss how this increase in processing capacity is actually used in simulations involving polarizable potentials.

Molecular dynamics simulations were carried out on a system of 216 water molecules. The SPC model geometry and Lennard-Jones parameters were used.[33] Values for the atomic polarizabilities were obtained from Thole[24] and are listed in Table II. The partial atomic charges were chosen to reproduce the gas-phase dipole moment[9] (Table III). The density of the system was maintained at 1

TABLE I.
System Specifications of Machines Used in This Study.

Machine	Number of Processors	Peak Speed, Mflops[a] Theoretical	Benchmarked[b]	Memory (Mb)
SGI 4D/340	4[c]	13[d]	9[d]	64
HP 9000/735	1	198	120	144
CM-5 (32 nodes)	32	4000	1900	1024
CM-5 (64 nodes)	64	8000	3800	2048

[a] From ref. 32.
[b] Using LINPACK and other linear equation solvers.
[c] Only one processor was used in our calculations.
[d] Estimated speed of one processor.

g/cm^3. The temperature was maintained at 298 K by applying the velocity rescaling approach of Berendsen and co-workers[34] with a relaxation time of 10 fs. Timesteps of 2 fs were used in the dynamics. Periodic boundary conditions were enforced, and a spherical molecular cutoff of 8.5 Å was used in evaluating the nonbonded interaction functions.

The times required to calculate the nonbonded interaction terms during a 500-step molecular dynamics simulation of water using the polarizable potential are shown in Table IV. The iterative approach with a convergence criterion of 0.01 D was used to calculate the induced dipole moments. Results for systems containing 60 and 216 water molecules are shown.

The time required to perform the calculation on the HP computer scales roughly as N^2, which is in agreement with the runtime-N relation referred to in the Theory section. The increase in runtime with system size is not nearly as dramatic for the CM-5 since an increase in the number of operations performed on the nodes does not necessarily result in a commensurate increase in the time required for these operations. For the 216-water system, simulations on the 32- and 64-node partitions of the CM-5 are executed 19 and 32 times faster, respectively, than those on the HP 9000/735. Note that the time required to complete 500 molecular dynamics steps for 216 water molecules on a 32-node CM-5 is on the order of several minutes. This is comparable to the time (0.26 h) required to perform a similar simulation on the SGI 4D/340 using a nonpolarizable potential. Thus, projects involving standard-size liquid-state simulations using polarizable potentials can now be carried out on a 32-node CM-5 partition on a routine basis.

We analyzed the performance of the procedures used to evaluate nonbonded (polarization, Coulomb, and Lennard-Jones) interactions on the CM-5. We found that the code runs at 7–10 Mflops/node on the smaller partition.[†] This is considerably slower than the optimum performance reported for this computer (> 60

[†] The exact Mflop rating depends on factors such as the version of the operating system and compiler software as well as the presence of runtime performance and error-checking codes. For example, when running the iterative method, our code calculates runtime statistics for the iteration process, checks for nonconvergence, and switches to the matrix inversion method when necessary.

TABLE II.
Atomic Polarizabilities.[a]

Atom	α, Å^3
O	0.862
H	0.514
N	1.105
C	1.405

[a] From ref. 16.

TABLE III.
Partial Atomic Charges.

Molecule	Atom	q, Electronic Charge
water	O	−0.66900
	H	0.33450
ammonia	N	−0.80520
	H	0.26840
trans-N-methyl-acetamide	C (C^α)	−0.27608
	H (H−C^α)	0.08962
	C (carbonyl)	0.40677
	O	−0.53472
	N	−0.31666
	H (amide)	0.25695
	C (*N*-methyl)	0.14617
	H (*N*-methyl)	0.01624

TABLE IV.
Times (Hours) Required to Compute Nonbonded Interaction Terms (Polarization, Coulomb, and Lennard-Jones) during Molecular Dynamics Simulations of Water.

Number of Water Molecules	SGI 4D / 340	HP 9000 / 735	CM-5 32 Nodes	CM-5 64 Nodes
60	1.0	0.27	0.036	0.028
216	—[a]	3.9	0.21	0.12

The polarizable potential was used, and the dynamics were performed for 500 timesteps.
[a]Machine has insufficient memory.

Mflops/node)[35] but is reasonable since interprocessor communication is unavoidable in several parts of the calculations. First, atomic coordinates must be broadcast over all nodes at each timestep. Second, the iterative procedure for determining dipole moments requires evaluation of the product $\mathbf{A}\boldsymbol{\mu}$. This involves spreading the dipole moments over the nodes, obtaining the products of matrix–vector elements, and executing a reduction step to obtain the resultant vector. All these operations are done at each iteration, several of which are necessary to attain convergence at each timestep. Third, reduction of a few arrays is necessary in order to obtain the nonbonded energies and forces.

COMPARISON OF METHODS FOR EVALUATING INDUCED DIPOLE MOMENTS

As mentioned earlier, induced dipole moments may be obtained by either an iterative approach, which yields approximate values for the dipole moments, or by matrix inversion, which gives exact values. Representative runtimes required to compute the dipole moments using these two methods are listed in Table V, and a more complete set of iteration times is shown in Figure 1. It can be seen that the runtime required for the iterative approach roughly scales as $-k \log \delta$, where k is some constant and δ is the convergence criterion [eq. (10)]. Each of the lines in Figure 1 can be envisioned as consisting of two straight segments (and thus two values of k), with one section existing at $\delta > 10^{-5}$ and the other at $\delta < 10^{-6}$. The existence of two segments can be explained by first noting that calculations with $\delta < 10^{-6}$ require more than 15 iterations. Beyond this point, the damping parameter b [eq. (9)] attains a large and constant value, leading to slower iterations and a larger value for k. On the other hand, calculations requiring fewer than 15 iterations use linearly increasing values of b.

The general relation between runtimes and δ is consistent with the documented rate of convergence of the Jacobi method[26] and is observed here for both serial and parallel machines. The value of k for the two CM-5 partitions is 40–60 times smaller than its value for the HP computer. This means that for the CM-5, the evaluation of $\boldsymbol{\mu}$ at any iteration step is faster due to parallel execution of the linear algebra operations specified in eqs. (4), (8) and (9). In other words, the processing of data within an iteration step is done in parallel. In contrast, the execution of iteration steps is still performed in a serial fashion (i.e., $\boldsymbol{\mu}_{m-1}$ must be evaluated before calculations for $\boldsymbol{\mu}_m$ can start). We

TABLE V.
Selected Runtimes (Seconds) Required to Calculate One Set of Induced Dipole Moments for a System of 216 Water Molecules.

Method	Convergence Criterion, D	HP 9000 / 735	CM-5 32 Nodes	CM-5 64 Nodes
Iteration	1.0×10^{-2}	18	0.6	0.3
Iteration	1.0×10^{-5}	37	1.1	0.6
Iteration	1.0×10^{-8}	106	2.9	1.7
Matrix inversion	(Exact)	5400	49	27

A more extensive set of times is shown in Figure 1.

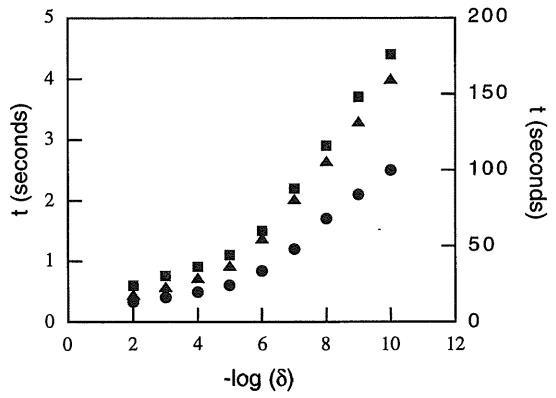

FIGURE 1. Runtimes required to calculate induced dipole moments via the iterative method. Times are shown as a function of $-\log \delta$. The squares represent runtimes obtained with the 32-node CM-5, and the circles those from the 64-node partition; in both cases the timescale on the left should be used. The (much larger) timescale on the right side of the figure is used for the runtimes (triangles) obtained with the HP 9000 / 735.

have opted to exploit parallelism in the processing of data rather than parallelism in the execution of the iteration steps[‡] since the amount of data is orders of magnitude greater than the number of iteration steps.

The times required for matrix inversion are considerably longer than the times required for the iterative method. However, there is a large difference between the matrix inversion runtimes for the two machines, even though the same algorithm (LU decomposition) was used on both the HP[26] and the CM-5.[36] This difference is due to the implementation of the LU routines on the parallel machine—the routines use blocking and load balancing in order to (1) perform operations on blocks of matrix elements rather than single data elements and (2) minimize the number of idle processors at any given time.[36] As a result, matrix inversion on the 32- and 64-node partitions of the CM-5 runs 110 and 200 times faster, respectively, than the analogous operation on the HP. For a system of 216 water molecules, a single matrix inversion step on the latter machine takes 1.5 h to complete.

[‡]For example, one can pipeline the execution of the iteration steps on a parallel machine. Suppose μ_i is evaluated on the ith processor. This processor need not wait for all the other induced dipole moments to become available before it can begin calculating a new value for μ_i. The process can start as soon as one of the other induced dipole moments becomes available and continue until contributions from all the other induced dipole moments have been evaluated.

It is clearly not practical to solve for $\boldsymbol{\mu}$ using the matrix inversion algorithm on the single-processor HP.

In the course of our simulations, we have encountered situations in which the iterative method fails to converge. In cases such as these, the program automatically switches to matrix inversion in order to obtain the correct induced dipole moments and continue the simulation. We have found that the faster iterative procedure may fail to converge when two polarizable sites are very close to each other. This configuration results in unusually large induced dipole moments on the two sites, and such values are difficult to find in the $3N$-dimensional space described by eq. (4). We have encountered such configurations rather infrequently during regular molecular dynamics simulations of polarizable water. These situations, however, are seen more often in free-energy perturbation (FEP) calculations which involve reduction of the Lennard-Jones parameters, which allows close encounters between polarizable sites. The evaluation of induced dipole moments in FEP simulations is discussed later.

USE OF POLARIZABLE POTENTIALS IN FREE-ENERGY PERTURBATION CALCULATIONS

Free-energy perturbation calculations are employed to determine the free-energy difference, ΔG, between two states. This method has been applied to calculate free energies of solvation, energy differences between conformers in solution, and a host of other chemical properties.[37-43] The free-energy difference between two systems a and b is given by

$$\Delta G_{ba} = -RT \ln\langle \exp[-(H_b - H_a)/RT] \rangle_a \quad (15)$$

where H_a and H_b are the Hamiltonians of states a and b, respectively; R is the gas constant; and T is the temperature. The ensemble average is taken over the state a, and system b is treated as a perturbation on system a. In practice, a and b represent significantly different chemical states, and it is necessary to perform the perturbation via several intermediate states. If we use the coupling parameter λ to denote a particular intermediate state

$$H_\lambda = (1 - \lambda)H_a + \lambda H_b \quad (16)$$

then the transformation from a to b can be accomplished in M steps ("windows") as

$$\Delta G_{ba} = \sum_{\lambda=0,\frac{1}{M},\frac{2}{M},\ldots}^{1} \Delta G_\lambda \quad (17)$$

At each window,

$$\Delta G_\lambda = -RT \ln\left\langle \exp\left[-(H_{\lambda+1/M} - H_\lambda)/RT\right]\right\rangle_\lambda \quad (18)$$

Equations (17) and (18) describe the windows method for performing FEP calculations. Each set of calculations typically involves 10–40 windows. Within each window, the system is first equilibrated to allow adjustment to the new H_λ. This is followed by sampling to obtain the ensemble averages in eq. (18).

An alternative to the windows method involves setting M to some much larger value (e.g., $M > 10^3$). This leads to miniscule incremental changes in H_λ. If one assumes that the system remains essentially in equilibrium, then eq. (18) can be approximated by

$$\Delta G_\lambda = H_{\lambda+1/M} - H_\lambda \quad (19)$$

Equations (17) and (19) form the basis for the "slow-growth" method.[§]

Equations (18) and (19) involve taking the difference of two relatively large numbers to obtain a small result. For our system [see eq. (1)],

$$H_{\lambda+1/M} - H_\lambda = (U_{tot})_{\lambda+1/M} - (U_{tot})_\lambda \quad (20)$$

If the perturbation does not involve any changes in the atomic coordinates, then the pairwise (Coulomb and Lennard-Jones) solvent–solvent interactions are constant. This large term can then be excluded from the calculations, leaving

$$H_{\lambda+1/M} - H_\lambda = (U'_{tot})_{\lambda+1/M} - (U'_{tot})_\lambda \quad (21)$$

where

$$U'_{tot} = (U_{pair})_{(solute-solute)} + (U_{pair})_{(solute-solvent)} + U_{pol} \quad (22)$$

where the subscripts refer to interaction energies between species. Unlike the pairwise term, the many-body polarization energy cannot be decomposed into interspecies and intraspecies components. As a result, U_{pol} usually makes the largest contribution to the value of U'_{tot}.[‖] In a typical simulation (e.g., a small organic molecule in a system of 216 water molecules), the values of U'_{tot} are large and on the order of 700–1000 kcal/mol, while ΔG_λ is usually less than 0.2 kcal/mol.[#] If we want the latter quantity to be within 10% of its true value, then the potential energies in eq. (22) must be evaluated with uncertainties less than 0.002%. The calculation of U_{pair} yields exact values. This means that any uncertainty in U'_{tot} arises from the iterative determination of U_{pol}, which in our model can comprise practically 100% of U'_{tot}. According to eq. (3), U_{pol} is a function of the electric fields and induced dipole moments. Since exact values of E_i^0 are calculated, any errors in the polarization energy are due to errors in the dipole moments, the uncertainties of which must also be less than 0.002%.

In our simulations, the magnitudes of the induced dipole moments are usually on the order of 0.2–1.0 D. The uncertainties of their values, when obtained via iteration, depend on the value of the convergence criterion δ, which must be less than $0.2\ \text{D} \times 0.002\% = 4 \times 10^{-6}$ D. This means that the value of δ recommended by Caldwell and coworkers[3] (0.01 D), while sufficient for structural studies,[1] is clearly much too large for the determination of induced dipole moments in FEP calculations.

It should be emphasized that our determination of δ is only an order-of-magnitude estimate that is based on simple error analysis. Its exact value will depend on several factors such as the number of windows, the length of the sampling period in a window, the value of ΔG_{ab} and its variation with

[§]There are sampling problems associated with both windows and slow-growth methods which are mentioned in D. A. Pearlman and P. Kollman, *J. Chem. Phys.*, **91**, 7831 (1989), and D. A. Pearlman, *J. Comp. Chem.*, **15**, 105 (1994). These problems do not affect the comparisons made in this section since we are not trying to compare the predictions of FEP simulations with experimental results. We are instead concerned with the self-consistency of results—namely, the reproduction of values obtained using an exact method (matrix inversion) by faster simulations which use approximations (iteration).

[‖]A special case arises when the atomic polarizabilities and charges of the solute become negligible. This situation is usually encountered in simulations involving mutations of the Lennard-Jones parameters of an uncharged solute. In such cases, $(U_{pol})_{\lambda+1/M} = (U_{pol})_\lambda$, and eq. (21) reduces to a difference in pairwise solute–solute and solute–solvent interaction energies.

[#]Values are typical for a simulation involving a small molecular solute in a box of 216 water molecules, with sampling performed over 20 windows.

λ, and the relative magnitudes of energies involved in a specific perturbation. In cases requiring extremely small values of δ, it may be necessary to resort to matrix inversion to eliminate errors which are inherent in the iterative approach.

One can demonstrate the effects of δ on ΔG_{ba} via a series of FEP simulations. A suitable test case is that of ammonia in water. The solvent system (a box containing 216 water molecules) and conditions were identical to those used in the earlier simulations on pure water. Polarizable sites were present on both solute and solvent. The gas-phase geometry and Lennard-Jones parameters used for the solute were taken from the literature.[25] Values for the atomic polarizabilities are listed in Table II. Partial atomic charges were chosen to reproduce the gas-phase dipole moment and can be found in Table III.

For test purposes, we chose to determine the free energy required to reduce the partial charges on all atoms of NH_3 by 25%. The perturbation was divided into 10 windows with 0.5 ps of equilibration and 0.5–1.0 ps of data collection in each window. A dynamics timestep of 2 fs was used. Calculations were carried out using values of $\delta = 0.01$ D and $\delta = 1 \times 10^{-6}$ D, and the results were compared to those obtained using matrix inversion,** which yields exact values of U_{pol}. To test further variations in the results, the free energies for both forward [ΔG_{ba}, for (fully charged) → (partly charged)] and backward [ΔG_{ab}] perturbations were computed. Results for all three cases are shown in Figure 2. The outcomes for the forward and backward perturbations obtained with $\delta = 0.01$ D exhibit considerable hysteresis (i.e., the two curves do not agree with each other). It is perhaps fortuitous that one result is close to that acquired via matrix inversion; the average of the two results yields a number (−1.8 kcal/mol) which is quite different from the one obtained with matrix inversion (−2.5 kcal/mol). Use of a smaller value for δ (1×10^{-6} D) yields results for ΔG_{ba} and ΔG_{ab} which are in much better agreement with each other and close to those obtained using matrix inversion.

It is clear that FEP calculations on systems similar to ours must use relatively small values of the convergence criterion δ. This has been demonstrated using the windows method. The slow-growth method has even more stringent requirements for δ, since this technique evaluates ΔG_{ba} in thousands of small increments ΔG_λ, each of which must be determined accurately. For example, if $\Delta G_{ba} = 20$ kcal/mol, then a 20,000-step calculation will require evaluation of increments which are on the order of 0.001 kcal/mol. These estimated values for ΔG_λ are at least two orders of magnitude smaller than those encountered using the windows method, and corresponding decreases in δ for the iterative calculation of induced dipoles are necessary.

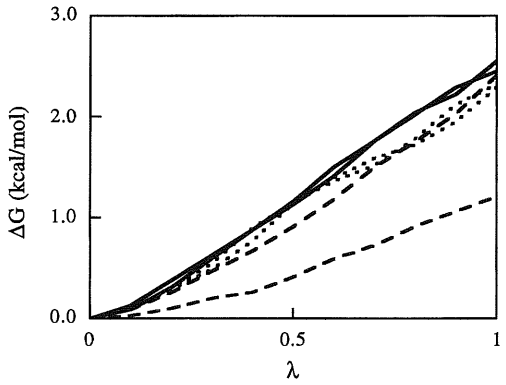

FIGURE 2. Comparison of ΔG required to reduce the partial atomic charges on NH_3 by 25%. The windows method was used. The two dashed lines represent the results of forward and backward perturbations using $\delta = 0.01$ D. Results obtained with $\delta = 1 \times 10^{-6}$ D are denoted by the two dotted lines. Results obtained using matrix inversion are represented by the solid lines. Uncertainties in the final values of ΔG are 0.3 kcal / mol for the last two sets of simulations and 1.3 kcal / mol for the first set.

The effects of δ on ΔG_{ba} obtained via the slow-growth method can be demonstrated. For this test calculation, we chose to determine the free energy required to reduce the partial atomic charges on *trans-N*-methylacetamide (NMA) by 25%. A box of 27 water molecules with periodic boundary conditions was observed, and the density and temperature were identical to those used in the earlier simulations. The geometry and Lennard-Jones parameters used for the solute were again taken from the literature.[25] Values for the atomic polarizabilities are listed in Table II. Partial atomic charges were chosen to best reproduce the gas-phase dipole moment and the electrostatic potential derived from *ab initio* calculations (Table III).[44]

Both ΔG_{ba} and ΔG_{ab} were determined using a 20,000-step slow-growth FEP calculation. Two sets of calculations using the iterative method with tolerances of 1×10^{-6} and 1×10^{-8} D were per-

**To our knowledge, this is the first time matrix inversion has been used to determine induced dipole moments in an extended ($> 10^3$ timesteps) molecular mechanics simulation involving a large ($> 10^2$ polarizable sites) system.

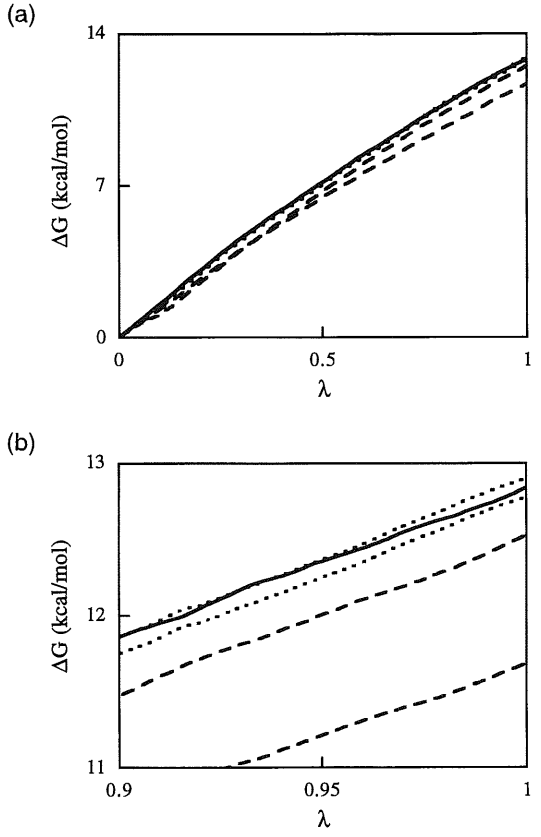

FIGURE 3. Comparison of ΔG required to reduce the partial atomic charges on *trans-N*-methylacetamide by 25%. The slow-growth method was used. The two dashed lines represent the results of forward and backward perturbations using $\delta = 1 \times 10^{-6}$ D. Results obtained with $\delta = 1 \times 10^{-8}$ D are denoted by the two dotted lines. Results obtained using matrix inversion are represented by the single solid line since results for both forward and backward calculations are virtually identical. (a) Results for the whole range of λ. (b) Results for $\lambda > 0.90$.

formed. Results for the forward and backward perturbations were compared to those obtained with the matrix inversion approach and are shown in Figures 3a and 3b. To obtain results which are in good agreement with those obtained using matrix inversion, it is necessary to use $\delta \leq 1 \times 10^{-8}$ D.

It is often the case that perturbations involving free-energy differences smaller than those in the preceding example are studied.[41,42] As a consequence, the values of ΔG_λ are often less than those of the NMA test case considered here, and even smaller values of δ are necessary in order to obtain reliable results using the slow-growth method. The need for highly accurate induced dipole moments in FEP simulations using slow growth may require use of the matrix inversion method.

Summary

In this study we have shown that polarization calculations can be performed on parallel computers in an efficient manner. On a 32-node CM-5 partition, our code runs at 7–10 Mflops/node, a speed which is reasonable since interprocessor communication is unavoidable in several parts of the calculation. Molecular dynamics simulations involving 216 water molecules are executed 19 times faster on the aforementioned partition than comparable runs on a single-processor HP 9000/735. Runtimes for standard-size liquid-state simulations using polarizable potentials on a 32-node CM-5 are comparable to those encountered when running simulations using nonpolarizable potentials on single-processor workstations. This performance is made possible by the proper distribution of data among the numerous processors, allowing evaluation of specific pairwise interactions and submatrices of the dipole tensor matrix on each node. Another advantage of computing on parallel machines is that matrix inversion becomes a practical alternative to the commonly used iterative method for calculation of induced dipole moments. Matrix inversion yields exact values for the induced dipole moments, which are important in certain situations such as those found in free-energy perturbation calculations. For these calculations, simple error analysis shows that induced dipole moments must have uncertainties of less than 0.002%. Such highly accurate determinations of induced dipole moments can only be obtained by (1) using a large number of iteration cycles with tolerances for convergence that are less than 10^{-6} D or (2) direct matrix inversion. The latter operation is executed 110 times faster on the 32-node CM-5 partition than on the HP 9000/735.

Acknowledgments

We thank Chris Marshall for help regarding the optimization of code for the CM-5. This work has been supported by grants from the National Science Foundation (DMB-9105208), the National Institutes of Health (GM30580), and the Columbia University Center for Biomolecular Simulations (NIH P41 RR06892) and by a grant of CM-5 time at

the National Center for Supercomputing Applications.

References

1. D. N. Bernardo, Y. Ding, K. Krogh-Jespersen, and R. M. Levy, *J. Phys. Chem.*, **98**, 4180 (1994).
2. P. Ahlström, A. Wallqvist, S. Engström, and B. Jönsson, *Mol. Phys.*, **68**, 563 (1989).
3. J. Caldwell, L. X. Dang, and P. A. Kollman, *J. Am. Chem. Soc.*, **112**, 9144 (1990).
4. A. Wallqvist, P. Ahlström, and G. Karlström, *J. Phys. Chem.*, **94**, 1649 (1990).
5. F. H. Stillinger and C. W. David, *J. Chem. Phys.*, **69**, 1473 (1978).
6. J. W. Halley, J. R. Rustad, and A. Rahman, *J. Chem. Phys.*, **98**, 4110 (1993).
7. U. Niesar, G. Corongiu, E. Clementi, G. R. Kneller, and D. K. Bhattacharya, *J. Phys. Chem.*, **94**, 7949 (1990).
8. M. Sprik and M. L. Klein, *J. Chem. Phys.*, **89**, 7556 (1988).
9. J. A. C. Rullman and P. Th. van Duijnen, *Mol. Phys.*, **63**, 451 (1988).
10. T. P. Straatsma and J. A. McCammon, *Chem. Phys. Lett.*, **177**, 433 (1991).
11. P. Barnes, J. L. Finney, J. D. Nicholas, and J. E. Quinn, *Nature* (London), **282**, 459 (1979).
12. J. E. Mertz, D. J. Tobias, C. L. Brooks III, and U. C. Singh, *J. Comp. Chem.*, **12**, 1270 (1991).
13. S. E. DeBolt, D. A. Pearlman, and P. A. Kollman, *J. Comp. Chem.*, **15**, 351 (1994), and references therein.
14. W. Smith, *Comp. Phys. Comm.*, **62**, 229 (1991).
15. B. R. Brooks and M. Hodošček, *Chem. Des. Automat. News*, **7**, 16 (1992).
16. T. W. Clark, J. A. McCammon, and L. R. Scott, In *Proceedings of the Fifth SIAM Conference on Parallel Processing for Scientific Computing*, SIAM, J. Dongarra, Ed., Society for Industrial and Applied Mathematics, Philadelphia, PA, 1992, p. 338.
17. J. F. Janak and P. C. Pattnaik, *J. Comp. Chem.*, **13**, 1098 (1992).
18. S. L. Lin, J. Mellor-Crummey, B. M. Pettit, and G. N. Phillips, Jr., *J. Comp. Chem.*, **13**, 1022 (1992).
19. R. Giles and P. Tamayo, *A Parallel Scalable Approach to Short-Range Molecular Dynamics on the CM-5*, Thinking Machines Technical Report, Thinking Machines Corp., Cambridge, MA, 1993.
20. D. Fincham, *Mol. Simul.*, **1**, 1 (1987).
21. W. S. Young and C. L. Brooks III, *J. Comp. Chem.*, **15**, 44 (1994).
22. S. Plimpton and B. Hendrickson, *J. Comp. Chem.*, submitted.
23. C. L. III Brooks, W. S. Young, and D. J. Tobias, *Int. J. Supercomp. Appl.*, **5**(4), 98 (1991).
24. B. T. Thole, *Chem. Phys.*, **59**, 341 (1981).
25. S. J. Weiner, P. A. Kollman, D. T. Nguyen, and D. A. Case, *J. Comp. Chem.*, **7**, 230 (1986).
26. W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes: The Art of Scientific Computing*, Cambridge University Press, New York, 1986, pp. 31, 652.
27. D. B. Kitchen, F. Hirata, J. D. Westbrook, R. M. Levy, D. Kofke, and M. Yarmush, *J. Comp. Chem.*, **11**, 1169 (1990).
28. Thinking Machines Corp., *The Connection Machine CM-5 Technical Summary*, Cambridge, MA, 1992.
29. Thinking Machines Corp., *CM Fortran Reference Manual*, Cambridge, MA, 1992.
30. D. A. Bader and A. Martin, *Performance of the CM-5*, Electrical Engineering Department Technical Report, University of Maryland, 1994.
31. E. A. Brewer and B. C. Kuszmaul, In *Proceedings of the 1994 International Parallel Processing Symposium*, H. J. Siegal, Ed., IEEE Computer Society Press, Los Alamitos, California, 1994, p. 858.
32. J. J. Dongarra, *Performance of Various Computers Using Standard Linear Equations Software*, Report CS-89-85, University of Tennessee, Knoxville, TN, 1994.
33. H. J. C. Berendsen, J. P. M. Postma, W. F. van Gunsteren, and J. Hermans, In *Intermolecular Forces*, B. Pullman, Ed., Riedel, Dordrecht, The Netherlands, 1981, p. 331.
34. H. J. C. Berendsen, J. P. M. Postma, W. F. van Gunsteren, A. DiNola, and J. R. Haak, *J. Chem. Phys.*, **81**, 3684 (1984).
35. L. Lane, R. Nandkumar, M. Heath, and L. Smarr, *Access* (National Center for Supercomputing Applications, Urbana–Champaign, IL), **8**, 4 (1994).
36. Thinking Machines Corp., *CMSSL for CM Fortran*, Cambridge, MA, 1992.
37. D. L. Beveridge and F. M. DiCapua, *Ann. Rev. Biophys. Chem.*, **18**, 431 (1989).
38. J. P. M. Postma, H. J. C. Berendsen, and J. R. Haak, *Faraday Symp. Chem. Soc.*, **17**, 55 (1982).
39. K. Ramnarayan, B. G. Rao, and U. C. Singh, *J. Chem. Phys.*, **92**, 7057 (1990).
40. W. Jorgensen, *Acc. Chem. Res.*, **22**, 184 (1989).
41. P. A. Bash, U. C. Singh, R. Langridge, and P. A. Kollman, *Science*, **236**, 564 (1987).
42. U. C. Singh, F. K. Brown, P. A. Bash, and P. A. Kollman, *J. Am. Chem. Soc.*, **109**, 1607 (1987).
43. T. P. Straatsma and J. A. McCammon, *Ann. Rev. Phys. Chem.*, **43**, 407 (1992).
44. J. D. Westbrook, Y. Ding, and K. Krogh-Jespersen, unpublished results.