Accelerated molecular dynamics simulation with the parallel fast multipole algorithm

John A. Board Jr., Jeffrey W. Causey, James F. Leathrum Jr.
Department of Electrical Engineering, Duke University, Durham, NC 27706, USA

Andreas Windemuth and Klaus Schulten
Beckman Institute and Department of Biophysics, University of Illinois, Urbana, IL 61801, USA

Received 30 June 1992

We have implemented the fast multipole algorithm (FMA) of Greengard and Rokhlin and incorporated it into the molecular dynamics program MD of Windemuth and Schulten, allowing rapid computation of the non-bonded forces acting in dynamical protein systems without truncation or other corruption of the Coulomb force. The resulting program speeds up simulations of protein systems with approximately 24000 atoms by up to an order of magnitude on a single workstation. Additionally, we have implemented a parallel version of the three-dimensional FMA code on a loosely coupled network of workstations, further reducing simulation times. Large (in both size of system and length of simulated time) protein molecular dynamics simulations are now possible on workstations rather than supercomputers, and very large protein computations are possible on clusters of workstations and parallel machines.

1. Introduction

Molecular dynamics simulation is a widely used technique for exploring the properties of atomic systems through explicit computation of particle motions. Though of use in many disciplines, molecular biology has benefitted particularly from the technique [1–4]. In a complex system like a protein with several tens of thousands of atoms embedded in a water or membrane environment, many kinds of forces are acting on the constituent atoms, including Coulomb, van der Waals, and various chemical bonding forces. Of these, the simple Coulomb force typically takes by far the longest to compute, as each atom in the system interacts electrostatically with every other atom in the system.

Straightforward evaluation of the Coulomb force for a system with n atoms is accomplished by summing over all pairs of particles at every timestep; this is an n-body problem with a computational complexity which grows as $\mathcal{O}(n^2)$. Though some workers truncate the range of the Coulomb field to reduce the computational complexity [5], this is thought to give poor energy evolution especially to large protein systems, where the long range effects of inhomogeneous charge distributions are thought to be important [6]. Others have reduced the runtime by large constants, but not in order, by observing that the forces acting between distant pairs of particles do not change significantly from timestep to timestep, so a hierarchy of simulation timesteps allows nearby particle pairs to update their force frequently, and increasingly distant particle pairs to update their forces less and less often [7]. Several groups [8–10] have felt the Coulomb (or equivalent gravitational) problem important enough to design and build custom parallel processors for rapid solution of the n-body problem.

Appel [13] and Barnes and Hut [11] developed "tree codes" which embed the computational region into a hierarchical tree structure [12]; these programs exploit the fact that a particle interacts with a distant group of particles much as if it were inter-

Correspondence to: K. Schulten, Beckman Institute and Department of Biophysics, University of Illinois, Urbana, IL 61801, USA.

acting with a single particle at the center of mass of the distant group. The tree structure (fig. 1) enables a systematic procedure to determine which particles are "distant" from each other; this algorithm reduces the complexity of the computation from $\mathcal{O}(n^2)$ to $\mathcal{O}(n \log n)$, though there is a loss in accuracy in the computation. Accuracy can be traded for compute time via a free parameter.

Greengard and Rokhlin [14,15] built on the tree code idea. They form the (infinite) multipole expansions for boxes on the lowest level of the tree (fig. 1) and carefully combine and shift these expansions as they are passed up and down the tree. Their algorithm reduces the complexity of the n-body problem to $\mathcal{O}(n)$, and the errors introduced by truncating the infinite multipole expansions to a finite number of terms are well understood; arbitrary accuracy can be assured a priori by retaining an appropriate number of terms in the expansions.

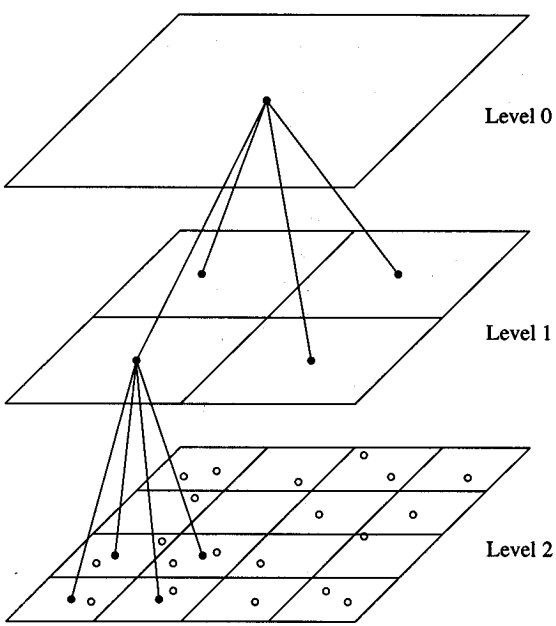

Fig. 1. Two-dimensional representation of the hierchical data structure used in both the tree codes [11–13] and the fast multipole algorithm [14]. The corresponding three-dimensional structure is an oct-tree. In the fast multipole algorithm, the particles are assigned to cells at the finest level of the tree; some cells may be empty, others may have several particles. The multipole expansion of the particle configuration in each box on the finest level is formed about the box center, each child box communicates this information to its parent box on the next level. Aggregate information about distant particles comes back down the three to low-level boxes.

2. Serial implementation

The fast multipole algorithm was implemented as a portable and well-structured program using the programming language C. The details this implementation are described elsewhere [16]. We have incorporated the fast multipole algorithm into the molecular dynamics program MD of Windemuth and Schulten [6]; this program implements the same force field for describing intramolecular interactions as the programs CHARMM [5] and XPLOR [17]. To our knowledge this is the first efficient application of the fast multipole algorithm to calculating the molecular dynamics of macromolecules.

Two modifications to the existing algorithms were done to combine the FMA program with the MD program. First, the FMA code was modified to also calculate the van der Waals forces for all atom pairs for which the interaction is calculated directly. This results in an irregular truncation of van der Waals forces, but at a distance at which the forces almost vanish due to their $\mathcal{O}(r^{-6})$ dependence on the distance between atoms r. Second, the MD program normally excludes non-bonding forces between atoms that are considered to be chemically bonded. Such exclusions could not easily be incorporated into the FMA algorithm itself. Instead, the program MD was modified to calculate the excluded forces only, which are then subtracted from the full forces calculated by the FMA. Since the excluded forces tend to be large, care has to be taken not to exceed the limits of numerical precision when subtracting them from the almost equally large total forces. We have found that the results are accurate enough even when using single precision numbers, so that numerical accuracy should be guaranteed at double precision arithmetic, which was used in all of the work reported here.

Benchmark results from this implementation obtained on a silicon graphics 4D/420 workstation are given in fig. 2. For the 24000 atom system the calculation time is reduced by an order of magnitude with only a small sacrifice in accuracy. This improvement will become more pronounced as even

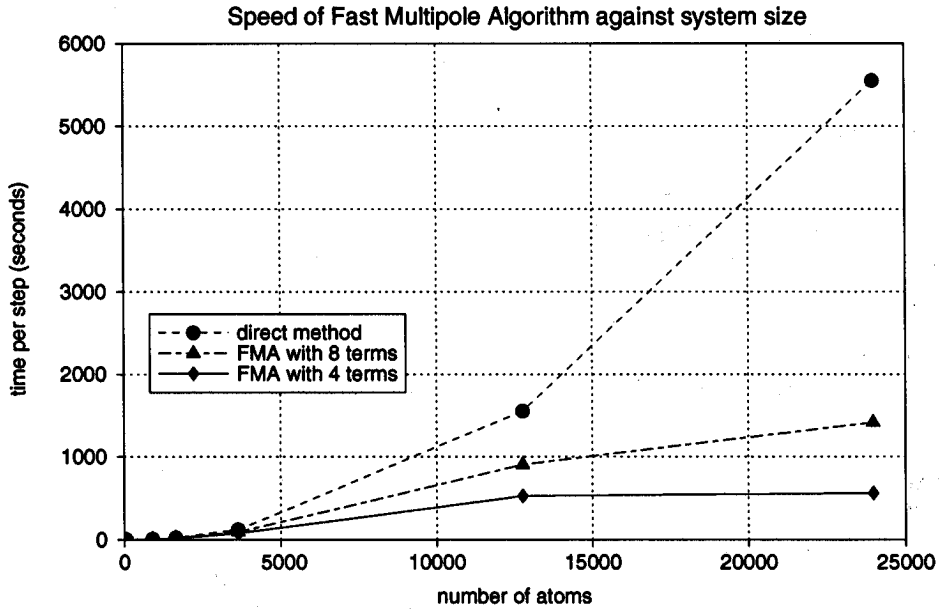

Fig. 2. Molecular dynamics benchmark with the fast multipole algorithm performed on an SGI 420 workstation. The non-bonded forces are calculated by direct pair summation and with the fast multipole algorithm at two different precisions. For the largest system, a 23975 atom bilayer assembly of lipids and water, an order of magnitude can be saved by using the FMA, with only a small loss of accuracy (see table 1).

Table 1
Performance of the fast multipole algorithm in molecular dynamics simulations of two large molecular systems. "Reaction center" is the photosynthetic reaction center of *Rhodopseudomonas viridis* consisting of 12637 atoms. "Membrane" is a patch of POPC lipid bilayer membrane with water caps on both sides and consists of 23978 atoms. "Terms" refers to the number of terms retained in the multipole expansions; "levels" refers to the number of levels in the tree data structure (fig. 1). The error is expressed as the relative deviation of the total Coulomb energy of the system as calculated with the fast multipole algorithm from an exact ($\mathcal{O}(n^2)$) computation, i.e. error $= |(e_{\text{exact}} - e_{\text{multipole}})/e_{\text{exact}}|$.

	Reaction center 12637 atoms		Membrane 23978 atoms	
	time (s)	rel. error	time (s)	rel. error
direct	457.8	0	1646.4	0
FMA				
3 levels, 4 terms	299.1	6.3×10^{-6}	337.8	3.5×10^{-4}
3 levels, 8 terms	327.4	2.1×10^{-6}	366.7	2.4×10^{-5}
4 levels, 4 terms	195.0	3.8×10^{-5}	179.4	1.4×10^{-3}
4 levels, 8 terms	514.7	2.1×10^{-5}	497.3	8.0×10^{-6}

larger systems are simulated. Table 1 shows the calculation times and energy accuracy for two different systems and two different truncation limits on an IBM RS6000 workstation. Retaining eight terms instead of four in the multipole expansions results in a significant increase in accuracy, but also more than doubles the calculation time. For molecular dynamics simulation the calculation speed is often the limiting factor in what problems can be solved. Therefore we believe that more than four terms are not

necessary for obtaining a valid molecular dynamics trajectory.

3. Parallel implementation

The serial implementation allows simulations that were previously only possible on large supercomputers to be performed by ordinary workstations. To take advantage of the method to solve problems that were previously inaccessible an implementation on the fastest available machines has to be attempted. The fastest machines available today are parallel computers, most of them of the *multiple instructions multiple data* (MIMD) type. We have implemented and tested a parallel version of the FMA (PFMA) using the coordination language Linda [18–20] on a network of workstations. Linda is a portable set of commands that is added to ordinary programming languages like C and FORTRAN to organize the data exchange and coordination between multiple threads of execution in a parallel program.

Linda is offered for a wide variety of parallel systems, ranging from networks of workstations to large massively parallel supercomputers. Networks of workstations generally have a lower communication bandwidth than dedicated parallel computers and are prone to disturbances from unrelated network traffic. Therefore, it is expected that the results reported here for a network of workstations will not only carry over to dedicated parallel machines, but even improve in performance. Implementations for the connection machine 5 and the intel paragon series of computers are planned.

Previously, two parallel implementations of the FMA have been reported, one by Greengard himself [21] and one by Zhao on the connection machine CM-2 [22]. None of these have been used in the context of molecular dynamics simulation, and the implementation by Greengard is for the two-dimensional version of the method only. The parallel, three-dimensional version of the FMA that was developed by us and which is the basis for the molecular dynamics implementation has been demonstrated to run efficiently in parallel on several machines (transputer, sequent symmetry, intel touchstone gamma, encore multimax) without the molecular dynamics code [16].

The basis for parallelisation is the separation of the simulation volume, i.e. the volume that is covered by the hierarchical grid used in the FMA, into a number of regions. An FMA calculation is then done for each region separately, omitting all calculations that do not affect the forces on the atoms of that region. It turns out that parallelization in this way is very efficient when the FMA is working in the multipole regime, i.e. in the regime where the calculation of multipole interaction dominates over the calculation of pair interactions. The quotient between the work done in the unseparated and in the task-separated FMA calculation is called the intrinsic efficiency. The intrinsic efficiency sets an upper limit on the speedup that can be achieved with this method on a given number of processors. Table 2 gives the intrinsic efficiencies for different systems, truncation limits and task separations. The parallel implementation reported here is based on a master/worker scheme, which provides automatic load balancing and makes the procedure suitable for heterogeneous networks, i.e. networks where some processors are able to do much more work than others. A master process running on one processor separates the FMA calculation into tasks and sends out one task per processor to the network. The remaining tasks are then assigned to the processors as they finish their previously assigned tasks, ensuring that the workload is as evenly distributed as possible. This scheme works best if the number of tasks is at least twice the number of processors.

The program MD with PFMA and Linda was tested by performing very short molecular dynamics runs on a network of five IBM RS6000 workstations. Due to a shortcoming in the implementation of the version of Linda used, one workstation has to be reserved as the central controller, making a maximum of four processors available for the actual PFMA calculation. Fig. 3 shows a benchmark of the 23975 atom system done with different task separations on a varying number of processors. The efficiency with four processors is $\approx 85\%$. 4% of the 15% loss is due to the intrinsic efficiency, which for this system is 96% (see table 2). The remaining loss is due to communications overhead and waiting times resulting from uneven task distributions. The latter is especially obvious for separation into 8 tasks, where a significant dip in efficiency is observed when using

Table 2
Intrinsic efficiency of parallel fast multipole algorithm for different configurations. The table shows that the task separation is more efficient for systems where the multipole–multipole interactions dominate, i.e. systems with a higher number of hierachical subdivisions (levels). Also, the efficiency increases with the number of terms used in the expansions and with the size of the system. As would be expected, the intrinsic efficiency is lowered when using a larger number of tasks, but this effect is surprisingly small.

	4 terms			8 terms		
	8 tasks	16 tasks	32 tasks	8 tasks	16 tasks	32 tasks
1638 atoms, 2 levels	0.54	0.52	0.5	0.57	0.56	0.54
1638 atoms, 3 levels	0.8	0.77	0.74	0.93	0.91	0.89
3634 atoms, 2 levels	0.51	0.5	0.48	0.52	0.51	0.49
3634 atoms, 3 levels	0.63	0.6	0.57	0.76	0.74	0.72
12637 atoms, 3 levels	0.54	0.54	0.54	0.56	0.56	0.56
12637 atoms, 4 levels	0.77	0.76	0.74	0.9	0.89	0.89
23975 atoms, 3 levels	0.64	0.55	0.5	0.66	0.57	0.52
23975 atoms, 4 levels	0.9	0.83	0.76	0.96	0.93	0.9

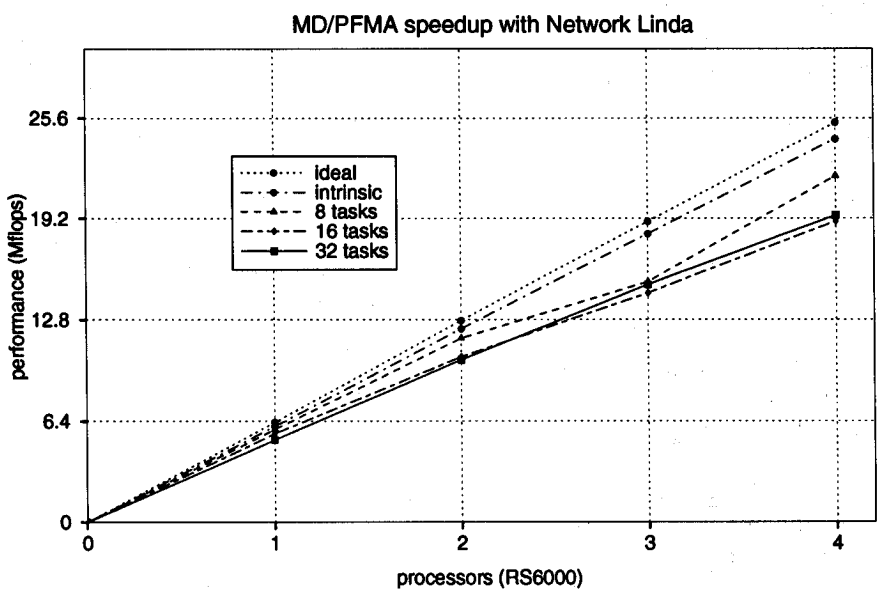

Fig. 3. Parallel molecular dynamics benchmark with fast multipole algorithm on one to four IBM RS6000 workstations. The intrinsic efficiency given here is for a separation into 8 tasks, the 16 task and 32 task intrinsic efficiencies are somewhat smaller (see fig. 2). The Mflops rating on the y axis is a performance measure related to molecular dynamics and is not identical to actual millions of operations per second.

three processors. This is explained by the fact that the distribution of 8 tasks to three processors leaves one processor idle a third of the time.

In fig. 3 the best result is obtained with a separation into 8 tasks, but this will change as the number of processors increases. For larger numbers of processors it is necessary to also use a higher number of tasks. The efficiency of the 32 task separation in fig. 3 shows almost no signs of levelling off; it can, therefore, be expected that the parallelization will still be efficient when a greater number (10–20) of workstations on a network are used to do a molecular dynamics calculation.

4. Discussion

With the incorporation of the FMA into the pro-

gram MD the feasibility of using the FMA in the context of molecular dynamics simulation of large macromolecular systems has been clearly demonstrated. The FMA is considered to be of tremendous importance in the molecular dynamics simulation of large, inhomogeneous macromolecular systems, where the use of cutoff schemes is questionable. It is especially promising in combination with distance class schemes [7], with which it is compatible. For the 24000 atom system, an increase in speed by two orders of magnitude can be achieved using a combination of these methods without sacrificing the long range interactions.

In addition to the enormous gain brought about by the serial implementation of the FMA we have also demonstrated the feasibility of carrying the method over to high performance parallel computing. The benchmark results from task separation and from actual parallel simulations on a small network of workstations suggest that good efficiency (on the order of 70%) can be achieved on machines with tens to hundreds of processors for the simulation of large macromolecular systems.

Acknowledgement

JAB and JFL were supported by the NSF. JWC was an undergraduate fellow of the Duke/NSF Engineering Research Center. AW and KS were supported by the National Institute of Health (grant P41RRO5969). We also thank the staff of the National Center for Supercomputing Applications for providing computational resources and the staff of Scientific Computing Associates, notably Tim Mattson, for generously providing advice about Linda.

References

[1] E. Clementi, G. Corongiu, M. Aida, U. Niesar and G. Kneller, in: MOTECC, Modern techniques in computational chemistry, ed. E. Clementi (ESCOM, Leiden, 1990) pp. 805–882.
[2] M. Karplus, in: Structure and dynamics of nucleic acids, proteins, and membranes, eds. E. Clementi and S. Chin (Plenum Press, New York, 1986) pp. 113–126.
[3] M. Karplus and J.A. McCammon, Ann. Rev. Biochem. 53 (1983) 263.
[4] M. Levitt and S. Lifson, J. Mol. Biol. 46 (1969) 269.
[5] B.R. Brooks, R.E. Bruccoleri, B.D. Olafson, D.J. States, S. Swaminathan and M. Karplus, J. Comput. Chem. 4 (1983) 187.
[6] A. Windemuth and K. Schulten, Mol. Simul. 5 (1991) 353.
[7] H. Grubmüller, H.Heller, A. Windemuth and K. Schulten, Mol. Simul. 6 (1991) 121.
[8] D.J. Auerbach, W. Paul and A.F. Bakkers. J. Phys. Chem. 91 (1987) 4881.
[9] K. Boehncke, H. Heller, H. Grubmüller and K. Schulten, in: Proceedings of the Third Conference of the North American Transputer Users Group, April 26, 27 (1990), Sunnyvale, CA. Transputer research and applications, Vol. 3, ed. A.S. Wagner (IOS Press, Amsterdam, 1990) pp. 83–94.
[10] D. Sugimoto, Y. Chikada, J. Makino, T. Ito, T. Ebisuzaki and M. Umemura, Nature 345 (1990) 33.
[11] J.E. Barnes and P. Hut, Nature 324 (1986) 446.
[12] L. Hernquist, Comput. Phys. Commun. 48 (1988) 107.
[13] A. Appel, SIAM J. Sci. Stat. Comput. 6 (1985) 85.
[14] L. Greengard, The rapid evaluation of potential fields in particle systems (MIT Press, Cambridge, MA, 1988).
[15] L. Greengard and V. Rokhlin, J. Comput. Phys. 73 (1987) 325.
[16] J.F. Leathrum Jr. and J.A. Board Jr., The parallel fast multipole algorithm in three dimensions. Technical Report, Duke University, Department of Engineering (1992).
[17] A.T. Brünger, in: Crystallographic computing, Vol. 4. Techniques and new technologies, eds. N.W. Isaacs and M.R. Taylor (Clarendon Press, Oxford, 1988).
[18] R.D. Bjornson, Ph.D. Thesis, Yale University, Department of Computer Science, New Haven, CT (1991).
[19] N. Carriero and D. Gelernter, Commun. ACM 32 (1989) 444.
[20] T.G. Mattson, R. Bjornson and D. Kaminsky, in: Proceedings of the Sun Users Group, Beyond parallelism to coordination, San Jose, CA (1991).
[21] L. Greengard and W.D. Gropp, Computers Math. Applications, 20 (1990) 63.
[22] F. Zhao, SIAM J. Sci. Stat. Comput. 12 (1991) 1420.