Parallel Molecular Dynamics: Communication Requirements for Massively Parallel Machines

Valerie E. Taylor

EECS Department
Northwestern University
Evanston, IL 60208

Rick L. Stevens   Kathryn E. Arnold

MCS Division
Argonne National Laboratory
Argonne, IL 60439

Abstract

Molecular mechanics and dynamics are becoming widely used to perform simulations of molecular systems, from large-scale computations of materials to the design and modeling of drug compounds. In this paper we address two major issues: a good decomposition method that can take advantage of future massively parallel processing systems for modest-sized problems in the range of 50,000 atoms and the communication requirements needed to achieve 30 to 40% efficiency on MPPs. We analyzed a scalable benchmark molecular dynamics program executing on the Intel Touchstone Delta parallelized with an interaction decomposition method. Using a validated analytical performance model of the code, we determined that for an MPP with a four-dimensional mesh topology and 400 MHz processors the communication startup time must be at most 30 clock cycles and the network bandwidth must be at least 2.3 GB/s. This configuration results in 30 to 40% efficiency of the MPP for a problem with 50,000 atoms executing on 50,000 processors.

1 Introduction

Molecular mechanics and dynamics are becoming widely used to perform simulations of molecular systems, from large-scale computations of materials to the design and modeling of drug compounds. While some small problems can be analyzed on high-end workstations, and in some cases on personal computers, important classes of problems require large parallel computers.

Classical molecular dynamics involve solving Newton's equations of motion for each atom in the system with respect to every other atom. The forces acting on each atom are a combination of forces due to bonds, Coulomb forces, and van der Waals forces. The calculation of the Coulomb forces is typically the most computationally intensive because each atom interacts electrostatically with all the atoms in the system. As an approximation many computational models compute interactions only within a certain radius of each atom; the cutoff radius reduces the number of interactions calculated so that the work scales linearly with the number of atoms. These short-range models have been shown to be very accurate for molecular dynamics. Other approximation techniques for N-body systems, such as the fast multipole method [8] and cell multipole method [5], show great promise for improving the run time of very large scale problems; however, they do little to help with small to modest-sized systems with at most 50,000 atoms, which is the focus of this paper.

We address two major issues: a good decomposition method that can take advantage of future massively parallel processing (MPP) systems and the communication requirements needed to achieve good efficiency. These issues are important to drug design and protein interactions, where researchers want to reduce the execution time of each time step for fixed-size problems. In these simulations the molecule of interest—often only a few thousands atoms—is modeled in a solution of water with many thousands of atoms. Reduction in execution time is important also in nonequilibrium molecular dynamics, where macroscopic changes may take significant time to evolve, requiring millions of time steps to model. Thus, for some problems, it is more significant to be able to execute many time steps on a modest-size problem than few time steps on a large-size problem. We analyze the use of current and future MPPs for these modest-sized problems.

We believe that computers designed and built in the next ten to twenty years will most likely fall into three general design families [14]. These three families comprise Cray-like systems with around 1000 processors, where each processor is roughly a teraflop in performance; mid-range systems with 10,000 to 100,000 processors, each with a performance of 10 to 100 gigaflops (an extrapolation from current MPP systems); and systems with up to 1,000,000 processors, each slightly faster than today's microprocessors and together reaching the petaflop level of performance. We are particularly interested in the last class of future machines because many believe it will be the first type to achieve petaflop performance. This class of machines is likely to be memory limited because of cost. Molecular dynamics is a good application to examine because it has modest memory requirements.

Many implementations of parallel molecular dynamics have been developed for the first two classes of MPPs [3, 4, 6, 7, 11, 16, 17], but few groups have addressed issues related to the use of the third class, particularly for small to modest-sized problems. In this paper we focus on a fine-grained decomposition of the molecular dynamics algorithm that parallelizes beyond the number of atoms in the systems. Traditional approaches use decompositions that partition the problem either in space or by particle (or atom). The parallelism available with these methods is limited by the number of atoms, excluding parallelizing over time. In contrast, we use an *interaction decomposition* method that assigns unique atom pairs to different processors; the pairs represent the interactions. This method is limited by the number of interactions, which is much greater than the number of atoms for short-range models.

Specifically, we extend the work of Plimpton [16] to consider the case of the number of processors greater than or equal to the number of atoms. We present an analytical performance model of real code, which allows us to understand the performance on existing machines and identify the features needed to achieve good performance on future machines. Our analysis indicates that for an MPP with a four-dimensional mesh topology and 400 MHz processors, the communication startup time must be at most 30 clock cycles and the network bandwidth at least 2.3 GB/s. This configuration results in 30 to 40% efficiency of the MPP for a problem with 50,000 atoms executing on 50,000 processors.

In the following section we provide details about the sequential benchmark program. In Section 3 we discuss our interaction decomposition scheme and analyze the communication and computation requirements for the parallel code. The analytical model is validated in Section 4. This model is extended in Section 5 to consider systems of up to 50,000 atoms executing on future MPPs with up to 40,000 Intel Delta-type processors and network. In Section 6 we investigate the communication requirements necessary to achieve good performance on MPPs. We summarize the paper in Section 7.

2 Background

The benchmark problem used in this work is a simulation of helium atoms at room temperature (300 K). The basic quantities associated with the molecular model are the number of molecules per unit volume and the mass, size, and velocity of each molecule. The effect of the molecules' interacting with each other at a particular temperature and pressure can be calculated from knowledge of the intermolecular force field. The force is effectively zero at large distances; it becomes weakly attractive when the molecules are sufficiently close, but decreases again to be strongly repulsive at short distances. Thus, the short-range model is appropriate for the gaseous class of problems.

The benchmark program uses a cutoff radius of 6.44 Angstroms, or 2.5σ, where σ is the distance at which the potential energy is equal to zero using the Lennard-Jones potential model. Intermolecular collisions in dilute gases, such as helium gases, are very likely to be collisions involving only two atoms. Hence, the actual force computations are executed only on atom pairs with a distance less than the cutoff radius; these computations comprise less than 4% of the total possible interactions. This value was derived by taking the ratio of the volume of the sphere with radius 6.44 Angstroms to the volume of the cube. This value was also validated by experimental results involving the counting of calculated interactions.

The sequential molecular dynamics algorithm is given in Table 1. The velocities are initialized to random values. The potential energy between pairs of helium atoms is calculated by using the Lennard-Jones (12-6) potential model. The derivative of the energy with respect to the distance is equal to the force between the atoms. Hence, the potential energy is then used to calculate the attractive and repulsive forces for each atom. The result is an $N \times N$ force matrix, $\mathbf{F}$, where N is the total number of atoms and F_{ij} is the force on atom i due to atom j. This matrix is sparse because of the short-range model and skew-symmetric ($F_{ij} = -F_{ji}$) because of Newton's third law. Based upon these forces, the atoms move within the cube.

We estimated the sequential computation cost by counting the number of different operations and ex-

Table 1: Molecular Dynamics Algorithm

1. Predict coordinates and velocities
2. Compute particle interactions
3. Calculate potential energy
4. Correct coordinates and velocities
5. Sum all velocities
6. Calculate kinetic and total energies
7. Determine temperature and scale velocities, if necessary

Repeat steps 1 through 7 for I time steps

perimentally determining the execution time of the different operations. The result of this count is given below.

Serial Computation Cost

$$
\begin{aligned}
&It_{abs} + It_{sqrt} + t_{exp} + (I+1)t_{log} \\
&+[3\sqrt[3]{N} + (63I+7)(2N) + (2I+3)(3N) \\
&+5I\tfrac{N(N-1)}{2} + 11I + 1]t_{add} \\
&+[3\sqrt[3]{N} + (51I+15)(2N) + (3I+3)(3N) \\
&+16I\tfrac{N(N-1)}{2} + 40I + 39]t_{multiply} \\
&+[22N + I\tfrac{N(N-1)}{2} + 4I + 32]t_{divide} \\
&+[(17I+15)(2N) + 3N + 8I\tfrac{N(N-1)}{2}]t_{subtract} \\
&+[(16I+21)(2N) + (2I+1)(3N) + 23I + 29]t_{equal}
\end{aligned}
$$

The term I represents the number of time steps and t_{op} the execution time of the specific operation. The term $\frac{N(N-1)}{2}$ corresponds to the total number of possible interactions, where only half of the interactions are calculated (because of Newton's third law).

The benchmark program does not use any techniques to reduce the number of atoms that must be checked for possible interactions; all atoms are checked. However, when we explore the case in which the number of processors is approximately equal to the number of atoms, we consider the reduction in computation that results from using such techniques. As we will see in Section 6, this reduction affects the communication requirements. We consider also the case in which the lattice increases with larger systems.

3 Interaction Decomposition

Conventional decomposition methods for molecular dynamics problems consist of assigning a partition of the particles or the physical space to the different processors. With these methods two atoms that are within the cutoff radius may be assigned to different processors. Particle decomposition requires global communication of particle coordinates among all processors to identify interactions to be calculated. Spatial decomposition requires local communication of particle coordinates among processors with adjacent physical regions; additional communication is required for atoms that move beyond the physical region assigned to a given processor.

In contrast to the conventional methods, our interaction decomposition method assigns unique particle pairs to different processors. The pair lists are the potential interactions to be calculated during the simulation. All possible interactions are assigned to processors. Hence, no interprocessor communication is needed to calculate any interactions.

In our code, the interactions are mapped to a two-dimensional mesh of processors in the following manner. Recall that each entry of the $N \times N$ force matrix, F_{ij}, represents the force on atom i from j. Divide this matrix into $\sqrt{P} \times \sqrt{P}$ equal-sized blocks. Each block has $\frac{N}{\sqrt{P}} \times \frac{N-1}{\sqrt{P}}$ or $\frac{N(N-1)}{P}$ interactions. Also recall that the force matrix is skew symmetric, for which only the interactions in either the upper or lower triangular part of the matrix need to be calculated. The remaining entries are known because $F_{ij} = -F_{ji}$. Consider the interactions in the upper triangular section. We assign half of the interactions in each block of a given processor in this section to its transpose processor; each processor is assigned $\frac{N(N-1)}{2P}$ interactions. Hence, each interaction is mapped to a unique processor; the mapping is similar to that of Plimpton [16].

An example of this mapping is illustrated in Figure 1 for a 4×4 array of processors with 16 atoms, numbered 1 to 16. Each box corresponds to a processor; the processor is identified by the (i, j) coordinate of the grid. The assigned interactions are given within each box. Consider dividing the particle list into $\sqrt{P}$ sections (or, in this case, four sections). The diagonal processors are assigned the interactions between the atoms in the respective sections. For example, processor (2,2) is assigned the interactions between atoms in section 2—atoms 5, 6, 7, and 8. The interactions between two sections of the atom list are split across two transpose processors. Consider processors (1,3) and (3,1). These two processors split the interaction calculations between section 1 and section 3 of the atom list. Processor (1,3) is assigned the interactions between atoms 1, 2, 3, 4 (section 1) and atoms 11, 12 (one half of section 3); processor (3,1) is assigned the interactions between atoms 1, 2, 3, 4 (section 1) and

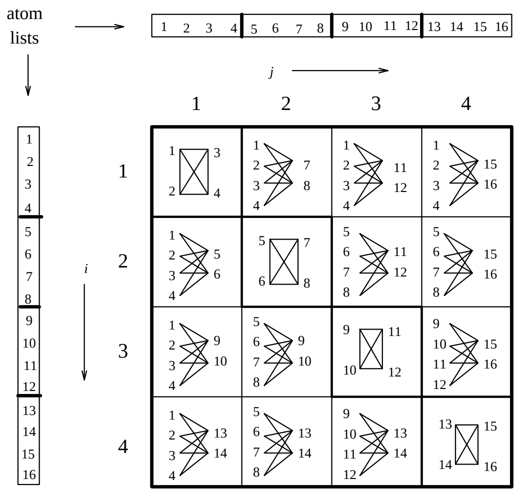

Figure 1: An example of mapping interactions for 16 atoms to 16 processors

atoms 9, 10 (the first half of section 3).

The communication pattern required for the force updates consists of communication in a row of processors, in a column of processors, and between transpose processors. This pattern results because each processor, P_{ij}, has atoms in common with the processors in the same row, row i, same column, column j, and the processors in the same row and column of the transpose processor, row j and column i. Hence each processor first communicates with the processors in the same row, then communicates with the processors in the same column. During these two communications, each processor keeps a list of information that is to be used by its transpose processor. After completing the two ring communications, this information is exchanged with the transpose processor. In this way processor P_{ij} has information from row i, row j, column i, and column j of processors.

The following is the cost associated with communicating the force updates between processors and updating the corresponding data structures.

<u>Interprocessor Communication Cost</u>

$$
\begin{aligned}
&[6I(\tfrac{3}{2}\tfrac{N}{\sqrt{P}}) + 12I(\tfrac{3}{2}\tfrac{N}{\sqrt{P}})(\sqrt{P}-1)]t_{add} \\
&+[(6I+2)(\sqrt{P}-1)]t_{multiply} \\
&+[9I(\tfrac{3}{2}\tfrac{N}{\sqrt{P}}) + (6I+2)(\sqrt{P}-1)]t_{divide} \\
&+[6I(\tfrac{3}{2}\tfrac{N}{\sqrt{P}})(\sqrt{P}-1) + (20I+1)(\sqrt{P}-1)]t_{equal} \\
&+[2(I+1)(\sqrt{P}-1) + 2I(\sqrt{P}-1) + I]t_{comm} \\
&+[I]t_{gsum}
\end{aligned}
$$

Table 2: Experimental execution times for the Intel Delta

t_i	Description	Delta (μs)
t_{abs}	fabs()	0.55289
t_{sqrt}	sqrt()	2.56527
t_{exp}	exp()	75.5789
t_{log}	log()	9.63801
t_{add}	addition	0.25123
$t_{multiply}$	multiplication	0.25121
t_{divide}	division	3.89710
$t_{subtract}$	subtraction	0.25123
t_{equal}	assignment	0.12572
$t_{startup}$	comm. startup	714.35
$t_{net_latency}$	network latency	0.425 per byte

The term $t_{gsum} = (2 * \log_2 P)t_{comm}$ is the time to execute a global sum and $t_{comm} = t_{startup} + t_{net-latency} * message\_size$ is the communication time. The term $(\sqrt{P}-1)$ represents the number of messages communicated among a row or column of processors corresponding to a ring communication structure. The message size is $O(\frac{3}{2}\frac{N}{\sqrt{P}})$, where $\frac{3}{2}\frac{N}{\sqrt{P}}$ is the number of atoms assigned to a processor.

4 Model Validation

We analyzed the performance of the benchmark molecular dynamics simulation executed on the Intel Touchstone Delta. The original program was rewritten in C with the p4 message-passing library developed at Argonne National Laboratory [12]. This library allows the code to be portable to other machines such as the Paragon, IBM SP, and CM-5. We developed analytical models of the sequential and parallel execution time based upon the costs described in the preceding sections. The parallel execution time is the sequential time divided by P, plus the communication cost, which includes the cost to update the additional data structures. The values of the parameter t_{op} were obtained experimentally and are given in Table 2. The timings correspond to single-precision operations between an array and a scalar with the result placed in an array. This computation is representative of the computations found in the code. The time for an addition operation corresponds to 10 clock cycles of the 40 MHz Intel i860XR processor; this represents memory latency in addition to computation time. The values for the t_{comm} expression were also determined exper-

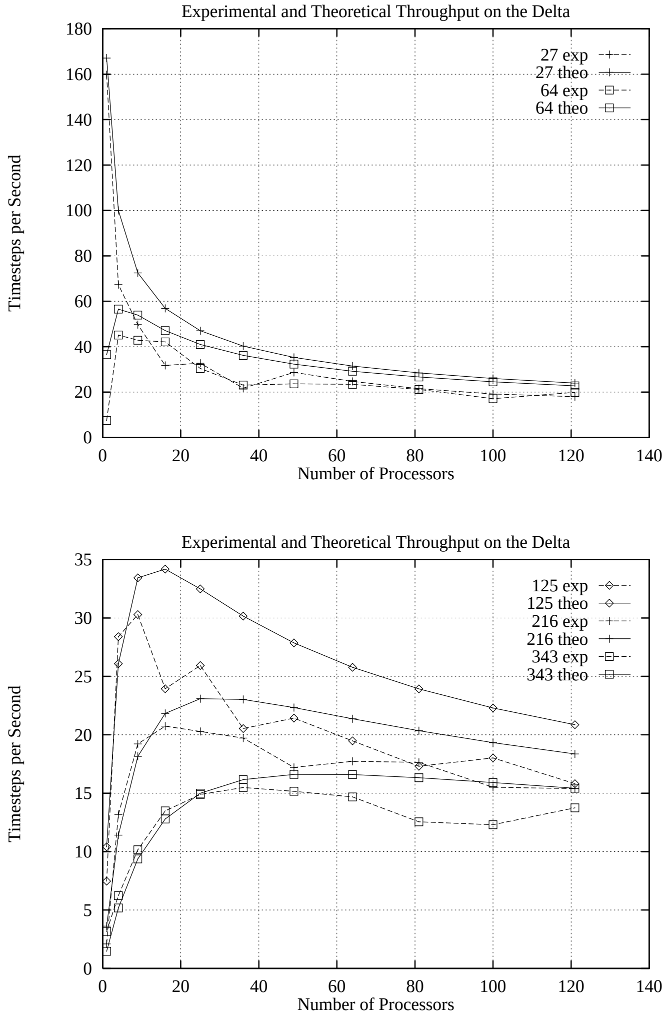

Figure 2: Theoretical and experimental throughput

imentally for the p4 library routines by using a least squares fit.

We considered problem sizes ranging between 64 and 343 atoms executing on 1 to 121 processors. These problem sizes allow us to explore the case of more processors than atoms on available machines. Further, the sizes fall within the range that is typical for the benchmark program. The results are given in Figure 2. The plots are in terms of throughput, or time steps per second; the throughput is equal to $\frac{1}{exec\_time} * 400$ (for the 400 time steps, considered as typical for an equilibrium type computation), where $exec\_time$ is the execution time in CPU seconds.

In all the plots the analytical model predicts the overall shape of the curves, especially for the 27- and 64-atom cases where we have more processors than atoms. There is a small difference between the theoretical and experimental results that we attribute to load balance synchronization, which our analytical models does not handle. The mapping described in Section 3 provides a straightforward communication scheme but does not balance the load among the processors; the initial mapping does not differentiate between interactions that are calculated and those that are eliminated by the cut off radius. Thus our mapping can result in load imbalance.

Given that the analytical model predicts the shape of the plots, we use this model to explore the communication requirements necessary to achieve good performance on future MPPs for modest-sized problems.

5 More Processors Than Atoms

We used a slightly modified version of our analytical model to predict what happens when we have more processors than atoms. Recall from Section 2 that the original code checks all atoms to determine the interactions to be calculated. In this section we consider an optimized code for which the number of computations is reduced by using techniques such as neighbor lists [18] and link-cell method [9]. The neighbor list technique consists of maintaining a list of nearby atoms for each atom. This list is examined for possible interactions instead of checking all atoms in the system; the list is rebuilt every few time steps. With the link-cell method, during each time step the atoms are binned into three-dimensional cells with side length equal to the cutoff radius. This reduces the task of finding neighbors to checking the atoms in 27 bins—the bin the atom is in and the 26 surrounding bins. Combined methods consist of keeping a neighbor list and using the link-cell method to update the neighbor list every few time steps.

Our modified model incorporates the combined method to reduce the calculations to check for possible interactions. We assume the neighbor list is updated every 20 times steps. In this way the cost of binning and checking the 27 bins is amortized over the time steps and is negligible, since it is only done 5% of the time. We considered problems where the lattice increases in size with an increase in the number of atoms but the cutoff radius remains the same; the number of neighboring particles remains a constant with different problem sizes. We assume the average size of the neighbor list is 78 atoms. This is consistent with the problems analyzed in [16].

Figure 3 provides the theoretical throughput for systems with 17,576 to 50,653 atoms executed on up to 40,000 processors. For systems in the range of

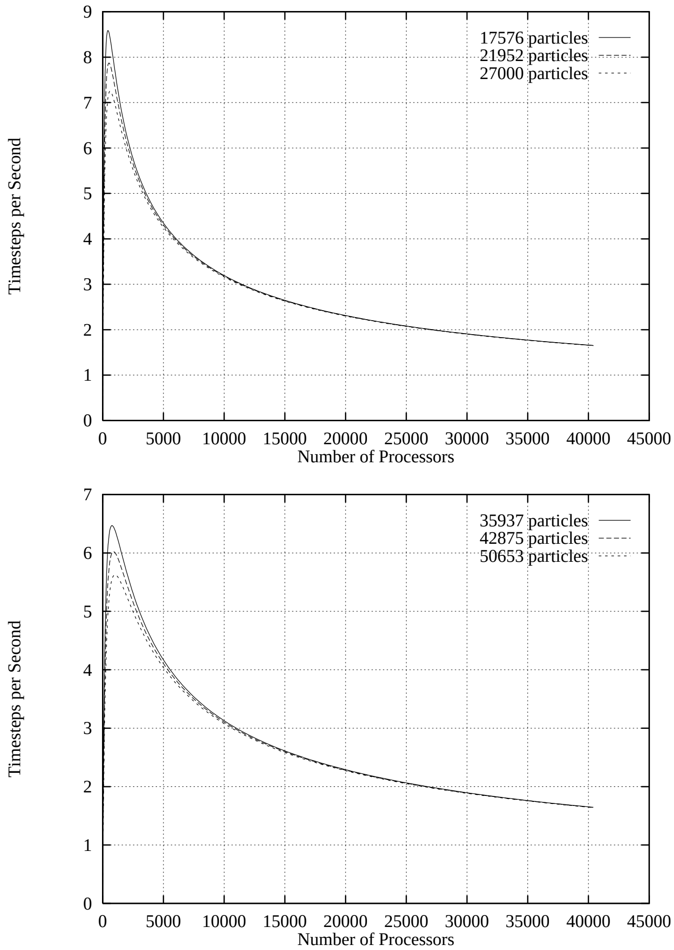

Figure 3: Theoretical performance on future MPPs

18,000 particles we see the throughput reaches a maximum with fewer than 500 processors; for systems in the range of 50,000 particles we see the maximum occurs around 1000 processors. Beyond this point the throughput begins to decrease significantly, and the efficiency drops down to approximately 0.1% for 40,000 processors. This analysis applies to the Intel Delta machine that has a clock frequency of 40 MHz and the network bandwidth for p4 messages of 2.3 MB/s.

6 Communication Requirements

In this section we explore the communication requirements necessary for future massively parallel machines to achieve efficiencies in the range of 30 to 40% when $P > N$. We consider the base machine with a clock that is an order of magnitude faster than the Delta processors, or 400 MHz. This results in the execution times of the various operations being an order of magnitude less, but the ratios between the various times remain the same. Hence all the values in Table 2 are reduced by an order of magnitude; the startup time is now 714e-7s and the network bandwidth is 23 MB/s. For illustrative purposes, we use the 50,653-atom problem and explore future MPPs with 40,000 to 60,000 processors.

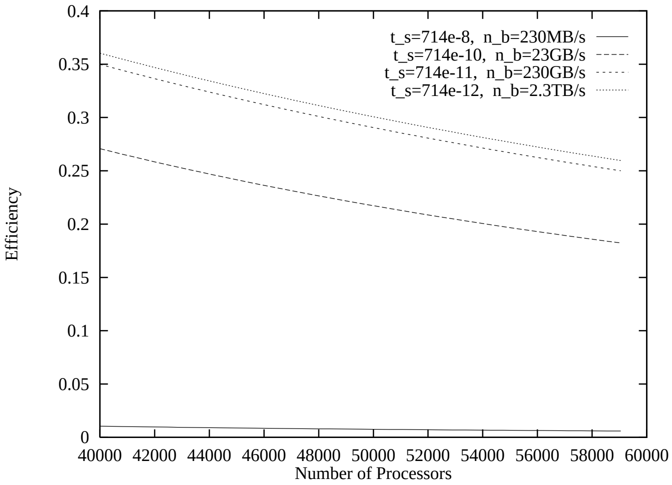

Figure 4: Efficiency plot for a 3-D mesh MPP for 50653 atoms

The communication requirements are affected by the communication startup time, the network bandwidth, and the machine topology. We explore the effects of all three parameters on performance. We assume the algorithm remains fixed. Figure 4 is the efficiency plot with varying startup times (identified by t_s) and network bandwidth (identified by n_b) for a three-dimensional mesh topology. The efficiency of the base machine (t_s=714e-7, n_b=23MB/s) is not shown on the graph because it is very small, close to zero. The point at which we achieve 30 to 40% efficiency is when the startup time is 714e-11s and the network bandwidth is equal to 230 GB/s. Further analysis indicate that 30 to 40% efficiency can also be achieved with the same startup time, 714e-11s, but a network bandwidth of only 2.3 GB/s. The startup time of 714e-11s corresponds to approximately 3 clock cycles for the 400 MHz processors. To achieve this startup time on future MPPs, we believe that a type of "active" message with significant hardware assistance is needed, similar to that found in the J-Machine [15].

Next we consider a four-dimensional mesh topology. Figure 5 is the efficiency plot with varying startup times and network bandwidth for this topology. The point at which we achieve 30 to 40% efficiency is when

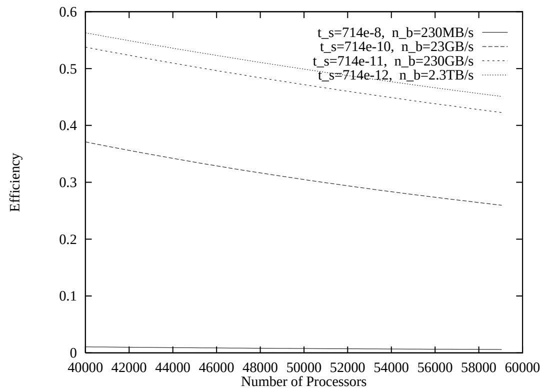

Figure 5: Efficiency plot for a 4-D mesh MPP for 50,653 atoms

the startup time is 714e-10 and the network bandwidth is 23 GB/s. Further analysis indicate that a 2.3 GB/s network suffices. This startup time corresponds to 30 clock cycles for the 400 MHz processors. We believe that this reduction can be achieved with a high-level language in the next ten to twenty years with little problem.

7 Summary

In this paper we addressed two main issues: a good decomposition method that can take advantage of a massively parallel system and the communication requirements needed to achieve good performance. Interaction decomposition, in contrast to partical and spatial methods, allows the scaling of processors beyond the number of atoms. Using a validated analytical performance model of some real code, we analyzed the use of MPPs for molecular dynamics applications. The analysis indicated that for an MPP arranged as a four-dimensional mesh with 400 MHz processors, the communication startup time must be at most 30 clock cycles and the network bandwidth at least 2.3 GB/s. We believe that the 30-clock-cycle startup time will be available to the user in a high-level language within a decade. This configuration achieves 30 to 40% efficiency of the MPP for a system with 50,000 atoms executing on 50,000 processors.

Acknowledgments

This work was supported by a National Science Foundation Young Investigator Award, under grant CCR–9357781, and the Office of Scientific Computing, U.S. Department of Energy, under Contract W-31-109-Eng-38.

References

[1] Barnes, J., and P. Hut "A Hierarchical O(NlogN) Force-Calculation Algorithm," *Nature*, 324:446, 1986.

[2] Bird, G. A., *Molecular Gas Dynamics*, Clarendon Press, Oxford, 1976.

[3] Brooks, B. R. and M. Hodoščck, "Parallelization of CHARMM for MIMD machines," *Chemical Design Automation News*, 7:16-22, 1992.

[4] Clark, T. W., J. A. McCammon, and L. R. Scott, "Parallel Molecular Dynamics," *Proc. 5th SIAM Conference on Parallel Processing for Scientific Computing*, 338-344. SIAM, 1992.

[5] Ding, Hong-Qiang, Naoki Karasawa, and William A. Goddard III, "Atomic Level Simulations on a Million Particles: The Cell Multipole Method for Coulomb and London Nonbond Interactions," *J. Chem. Phys.*, 97:4309-4314, September 1992.

[6] Fincham, D., "Parallel Computers and Molecular Simulation," *Molecular Simulation*, 1:1-45, 1987.

[7] Fox, G. C., M. A. Johnson, G. A. Lyzenga, S. W. Otto, J. K. Salmon, and D. W. Walker, *Solving Problems on Concurrent Processors: Volume 1*, Prentice Hall, Englewood Cliffs, New Jersey, 1988.

[8] Greengard, L., and V. Rokhlin, "A Fast Algorithm for Particle Simulations," *J. Comp. Phys.*, 73:325-348, 1987.

[9] Hockney, R. W., S. P. Goel, and J. W. Eastwood, "Quiet High-Resolution Computer Models of a Plasma," *J. Comp. Phys.*, 14:148-158, 1974.

[10] Karplus, M., and R. N. Porter, *Atoms and Molecules*, W. A. Benjamin, Inc., MenloPark, California, 1970.

[11] Lin, S. L., J. Mellor-Crummey, B. M. Pettit, and G. N. Phillips, Jr., "Molecular Dynamics on a Distributed-Memory Multiprocessor," *Comp. Chem.*, 13, 8:1022-1035, 1992.

[12] Lusk, E. and R. Butler, "User's Guide to the p4 Parallel Programming System," *Technical Report ANL-92/17*, Argonne National Laboratory, October 1992.

[13] Moss, D. S., and T. P. Flores, "Molecular Dynamics of Protein Molecules," *Supercomputational Science*, 251-268, 1990.

[14] *NASA-JPL Petaflops Workshop*, February 22-24, 1994, Pasedena, California.

[15] Noakes, M. O., and W. J. Dally, "System Design of the J-Machine," *Proceedings of the Sixth MIT Conference on Research in VLSI*, MIT Press, 179-194, 1990.

[16] Plimpton, S., "Fast Parallel Algorithms for Short-Range Molecular Dynamics," *Technical Report SAND91-1144*, Sandia National Laboratories, May 1993.

[17] Smith, W. "Molecular Dynamics on Hypercube Parallel Computers," *Physics Communications*, 62:229-248, 1991.

[18] Verlet, L., "Computer Experiments on Classical Fluids I: Thermodynamical Properties of Lennard-Jones Molecules," *Phys. Rev.*, 159:98-103, 1967.