Computational Limits of Classical Molecular Dynamics Simulations

Steve Plimpton*
Parallel Computational Sciences Department
Sandia National Laboratories†
Albuquerque, NM 87185-1111

15 May 1995

Abstract

The system sizes and time scales accessible by classical molecular dynamics techniques on current–generation parallel supercomputers are briefly discussed. The implications for simulation of glasses and glass–forming materials now and in the near future are highlighted.

1 Introduction

Classical molecular dynamics (MD) is a widely–applied simulation technique for studying the thermodynamic properties of liquids and solids [1]. As the workshop papers in this journal issue indicate, it has a particularly rich history as a tool used to probe the intricacies of the glass transition in a variety of ideal and real materials. Because an MD simulation computes real–space trajectories of individual atoms, it can yield detailed information on phase transitions and the temporal evolution of a glassy material. While running a large–scale glass simulation is a computational challenge, often the more difficult task is to sift through the immense volume of output data and identify interesting events such as diffusional hops over energy barriers or the internal (re)arrangement of order and disorder in localized clusters of atoms.

In the last decade, the advent of parallel supercomputers has dramatically increased the system sizes and time scales that can be studied with MD. However, there will always be practical limits to what can be simulated with existing algorithms with a reasonable investment of computer time. The purpose of this

*E–mail address: sjplimp@cs.sandia.gov
†Sandia is supported by the U. S. DOE under Contract DE–AC04–94AL85000.

short note is to highlight what is possible on today's and the near future's machines, and also to indicate where the bottlenecks are from a computational standpoint. The goal is to provide the materials researcher with a few "rules of thumb" to assist in assessing the feasibility of a proposed MD simulation.

2 Parallel MD

The dominant computational task in a MD simulation is to evaluate the forces acting on each atom at each timestep. In the classical MD models commonly used for simulations of glasses and glass–forming materials, atoms interact via simple, empirical, pairwise energy potentials — e.g. the 6/12 Lennard–Jones (LJ) potential [1]. Typically, such potentials are truncated so that atoms interact only with neighboring atoms within a short cutoff distance r_c. If we assume the simulated system has a nearly uniform atom density, then each atom interacts with a small, roughly constant number of neighbors. This means the computational effort per timestep scales linearly as N, the number of atoms simulated. Thus on a parallel machine with P processors, the best a parallel algorithm for short–range MD can scale is as N/P.

MD algorithms for parallel computers take advantage of the fact that the forces on different atoms and their subsequent motion can be computed independently and simultaneously by different processors. Various algorithms have been designed to exploit this natural parallelism [2, 3, 4, 5, 6]. The most widely used is known as *spatial–decomposition*. In this method the physical simulation domain (i.e. the volume) is split into sub–domains, one per processor. Each processor computes forces on only the atoms in its sub–domain and atoms migrate to new processors as they move from one sub–domain to the next. If we again assume uniform atom densities, this decomposition divides the computational work evenly across processors so that it scales as N/P. Inter–processor communication need only take place between timesteps to ensure each processor knows the updated positions of nearby atoms that exert force on the atoms within its sub–domain. In the limit of large numbers of atoms per processor, the communication cost scales as the $(N/P)^{2/3}$ surface–to–volume ratio of the sub–domains. Thus the overall algorithm scales optimally as the N/P computation. In practice, parallel efficiencies (the fraction of the optimal N/P speed-up actually realized) of 90% or more are often achieved in large simulations using spatial–decomposition methods [2, 5].

Despite this inherent parallelism, large–scale MD simulations are still computationally challenging. Consider that computing a LJ interaction between two atoms in 3–d requires about 25 floating–point operations (flops). At liquid/solid densities of $\rho^* = 0.75$–1.0 (in reduced units) with standard force cutoff lengths of $r_c = 2.5\sigma$ (where σ is the LJ unit of length), each atom interacts with 50–65 neighbors. In simulations of glassy states the vibrational motion of individual atoms must be accurately tracked, so the MD timestep is constrained to be a fraction of the Debye frequency for the material, and is typically on the order of a few femtoseconds. Thus to simulate a cube of glassy material a micron on a side (10^{10} atoms) for a few nanoseconds requires 10^{19} flops. Even efficient MD codes typically run at only about 10% of the peak flop rate on today's supercomputers (counting only LJ interaction flops), so a teraflop supercomputer would require 3 years to run this simulation! With this sobering realization we address what is currently feasible in the next section.

3 Performance Data

Table 1 lists the execution time for several supercomputers [7] on a standardized MD benchmark of a 3–d Lennard–Jones (LJ) system with a reduced density of $\rho^* = 0.8442$ and a force cutoff distance of $r_c = 2.5\sigma$. Each atom interacts with about 55 neighbors. The single processor ($P = 1$) Cray Y–MP and C90 timings are for a completely vectorized algorithm due to Grest et al. [8]; the entries marked with an asterisk are extrapolations of the linear scaling of the vector code. The parallel timings on 256 to 1904 processors of the Cray T3D and Intel Paragon were obtained with a spatial–decomposition algorithm. Both the benchmark and parallel algorithm are described fully in [5].

The large N parallel timings in the table exhibit near–optimal scaling; as the number of processors is doubled, the run times decrease by nearly a factor of two. For small N this is not the case; the parallel machines are still competitive with vector supercomputers, but there is a performance penalty for simulating a small system on large numbers of processors. Some additional comments on Table 1 may be useful:

- Because the spatial–decomposition approach is scalable, these numbers can be used to estimate timings for simulations on a smaller machine. For example, dividing both the N and P values for a particular entry by 4 (or any other value) should yield a nearly identical timing.

- Like most benchmarks, these codes are streamlined. A production MD code will run slower due to I/O or on–the–fly analysis. The Paragon numbers listed in the table are single precision; double–precision timings are about 20% slower. On the other hand, if one is willing to implement machine–specific optimizations (e. g. assembly code the force interaction, or steal cycles from the communication co–processor on the Paragon) these numbers can be improved upon. For example, on the Paragon, these optimizations yielded nearly a factor of 3 speed–up for large problems [5].

- The numbers in Table 1 represent the fastest MD timings we are aware of on the listed machines, but other large parallel machines such as the TMC CM–5 and IBM SP–2 achieve similar performance levels. A compendium of parallel MD benchmark timings is presented in [2]. The high parallel efficiencies of spatial–decomposition algorithms enable accurate predictions of performance even if large numbers of processors are not available. For example, optimizing and timing this MD benchmark on one IBM SP–2 processor for an N–atom simulation should provide a good estimate (within 10–20%) of how fast a problem with $256 * N$ atoms could run on a 256–processor SP–2.

- The size of systems that can be simulated is limited only by machine memory for storing vectors of atom positions and velocities and (if used) neighbor lists. Simulations with up to 600 million and 400 million atoms on the TMC CM–5 and Intel Paragon respectively are discussed in [2, 9].

- The number of interactions per atom computed in MD simulations similar to this LJ benchmark scales linearly with the reduced density ρ^* and as the cube of the force cutoff distance r_c. If only repulsive interactions are modeled ($r_c = 2^{1/6}\sigma$) or a 2–d simulation is performed, the run times can be reduced significantly.

- More complex interatomic potentials are more costly to compute. If pairwise potentials are used, they can often be pre–computed and stored in tabular form. A table–lookup implementation may then run nearly as fast as Lennard–Jones potentials. Many–body formulations such as three–body potentials for silica are more compute–intensive, though if they are short–range they still offer scalable N/P performance within the spatial–decomposition framework. True long–range potentials such as untruncated Coulombic interactions are considerably more expensive to use, though multipole approximations reduce their complexity to $O(N)$ [10] and they can also be parallelized within the spatial–decomposition context [11].

- If simple models can be used, MD simulations of molecular systems such as polymer chains can be implemented with little additional overhead. For example, the bead–spring model used by Kremer and Grest [12] to simulate polymer melt dynamics requires only one spring–force computation per monomer (to connect pairs of adjacent monomers) in addition to the usual LJ interactions with nearby monomers. However, in polymer systems the diffusion coefficient of the center–of–mass of each chain may decrease as some power of the inverse of the chain length; consequently, a few nanosecond simulation may capture less "physics" than in the equivalent monomeric system.

We listed timings in Table 1 that span a wide range of system sizes, since there are challenging glass simulation questions that can be formulated either as short–timescale runs of large–system simulations or as long–timescale runs of modest–sized systems [13]. In this spirit, in Table 2, we convert the per–timestep numbers to CPU hours for a million–step simulation of three system sizes from small to large. These numbers indicate that even on the largest supercomputers it is still a demanding computational task to simulate a million atoms for a million timesteps (few nanoseconds of real time).

4 The Future

What promise does the future of massively parallel computing hold for classical MD? Currently, the largest machines available are roughly 100 gigaflop (peak speed) computers with tens of gigabytes of memory — e. g. the Intel 1840–node Paragon, Cray 512–node T3D, TMC 1024–node CM–5, and IBM 512–node SP–2. In the next two years it is likely there will be a teraflop–class machine built. On this machine certain calculations such as large, dense matrix inversions will run at a speed of one teraflop (per second). While MD simulations will run at a fraction of that flop rate, they will be 10 times faster than the timings discussed above. By the turn of the century a 10 teraflop machine is predicted to be available with 100 times the memory of today's parallel supercomputers. Unfortunately, such a machine will cost tens of millions of dollars and available time on it for MD (or any other) simulations will be limited, just as CPU cycles on vector supercomputers are rationed today. As the article by Angell [13] indicates, this added computational power will mean certain glass simulations become feasible, but other questions will remain out of reach. Thus, even with more powerful computers the challenge to the theoreticians and simulators will be the same — to compensate for the limited size and time scales addressable by MD with new insight and more efficient computational algorithms.

References

[1] M. P. Allen and D. J. Tildesley, *Computer Simulation of Liquids*, (Clarendon Press, Oxford, 1987).

[2] D. M. Beazley, P. S. Lomdahl, N. Gronbech–Jensen, R. Giles, and P. Tamayo, *Annual Reviews of Computational Physics*, edited by D. Stauffer, vol 3, (World Scientific, Singapore, 1995), in press.

[3] D. Fincham, Molecular Simulation **1**, 1 (1987).

[4] S. Gupta, Comp. Phys. Comm. **70**, 243 (1992).

[5] S. J. Plimpton, J. Comp. Phys. **117**, 1 (1995).

[6] W. Smith, Comp. Phys. Comm. **62**, 229 (1991).

[7] The simulation results presented in Tables 1 and 2 were obtained on a Cray Y–MP and Intel Paragon at Sandia National Laboratories and a Cray C90 and T3D at Cray Research.

[8] G. S. Grest, B. Dünweg and K. Kremer, Comp. Phys. Comm. **55**, 269 (1989).

[9] Y. Deng, R. A. McCoy, R. B. Marr, R. F. Peierls, and O. Yasar, in *Proceedings of High Performance Computing 1995*, published by the Society for Computer Simulation, p. 95.

[10] L. Greengard and V. Rokhlin, J. Comp. Phys. **73**, 325 (1987).

[11] H. Q. Ding, N. Karasawa and W. A. Goddard III, J. Chem. Phys. **97**, 4309 (1992).

[12] K. Kremer and G. S. Grest, J. Chem. Phys. **92**, 5057 (1990).

[13] C. A. Angell, in this issue.

Problem Size	Cray Y-MP	Cray C90	Cray T3D		Intel Paragon	
N	P=1	P=1	P=256	P=512	P=1024	P=1904
500	.00930	.00373	.00252	.00245	.00327	.00327
2048	.0369	.0154	.00432	.00372	.00379	.00379
6912	.106	.0425	.00765	.00568	.00651	.00528
16384	.250	.103	.0121	.00747	.00813	.00610
50000	.733	.286	.0289	.0167	.0174	.0125
100,000	1.47	.592	.0509	.0293	.0282	.0171
1,000,000	14.7*	5.92*	.405	.205	.199	.113
10,000,000	147*	59.2*	---	1.85	1.77	.961
100,000,000	1470*	592*	---	---	---	9.11

Table 1: CPU seconds/timestep for MD simulations of various numbers N of LJ atoms on several vector and parallel supercomputers.

Machine/Processors	N = 6912	N = 100,000	N = 10^6
Y-MP/1	29.4	408	4083
Paragon/1024	1.8	7.8	55.3
T3D/256	2.1	14.1	112.5

Table 2: CPU hours to perform a 10^6 timestep MD simulation of LJ atoms using the timings in Table 1.