Obtaining Scalable Performance from Molecular Dynamics Codes on HPC Machines

Peter Coveney, Fabrizio Giordanetto
Queen Mary College, London

Neil Stringfellow
University of Manchester

May 31, 2002

Abstract

A large amount of computational time allocated on high performance computers is used for molecular dynamics simulations, and there is increasing demand for scalable codes particularly in the field of large molecule simulations in disciplines such as biochemistry. This paper investigates the performance of the latest generation of molecular dynamics codes on high performance machines and discusses design strategies which have led to increased scalability. Examples are given which demonstrate the ability of these codes to solve problems which scale in size with the number of processors available, as well as their scalability in terms of decreasing execution time.

1 Molecular Dynamics

Molecular Dynamics (MD) is a computational method that calculates the time dependent behaviour of a molecular system. The technique was developed in the 1950's by Alder and Wainwright with the first protein simulations appearing in the late 1970's. Today widespread biological applications of MD include simulation of solvated proteins, protein-DNA complexes and lipid systems investigating detail such as the thermodynamics of ligand binding and the folding of proteins.

Molecular Dynamics simulations generate information at the microscopic level, including atomic positions and velocities. Statistical mechanics is then used to convert this to macroscopic values of interest such as pressure, energy or more specific information such as the energetics of a conformational change or the binding free energy of a particular drug (the mathematical expressions that allow us to do this can be found in detail in many texts such as that by Allen and Tildesley [3]). The advantage MD has over similar techniques such as Monte Carlo simulation is that MD not only allows access the thermodynamic properties but to time-dependent phenomena too.

The molecular dynamics simulation method is based on Newton's second law, or the equation of motion $F = ma$, where F is the force exerted on the particle, m is its mass and a is its acceleration. From a knowledge of the force on each atom, it is possible to determine the acceleration of each atom in the system. Integration of the equations of motion then yields a trajectory that describes the positions, velocities and accelerations of the particles as they vary with time. From this trajectory, the average values of properties can be determined. Molecular dynamics simulations

can be time consuming and computationally expensive, however computers are getting faster and cheaper and the use of massively parallel machines can greatly extend the time scale and the molecular complexity that can be studied.

2 Molecular Dynamics Codes

Established molecular dynamics codes represent a large amount of compute time allocated on high performance computing (HPC) machines. Current codes which are used include Amber, Gromacs and DL_POLY, but as new machines increase in speed and size the scalability of these codes on large numbers of processors is an important factor in determining the size of problem which can be run. Discussions of the nature of molecular dynamics codes [1] suggest that these codes may not scale well for large numbers of particles as well as numbers of processors on cache based machines.

There are many molecular dynamics codes available for scalar and parallel systems including Amber, DL_POLY, Gromacs, LAMMPS and NAMD, but for high performance computing the main priorities for deciding on a suitable code is the speed of execution and scalability of the executable. For biochemical problems in a HPC environment, the issue of scalability is not only restricted to speed-up over a large number of processors, but also the ability of a code to cope with very large numbers of atoms in a simulation.

Codes such as Amber and DL_POLY, in common with older Molecular Dynamics codes, employ the replicated data model, which requires all atom data to be replicated on all processors. Codes which have been written very recently, such as NAMD and LAMMPS, have parallel execution as a major design specification, and use spatial domain decomposition techniques, so that the amount of atom data stored on each processor scales as $1/p$, where p is the number of processors. For problems on many processors, a problem with n atoms would therefore require only n/p sets of atom data to reside on each processor, and whilst larger amounts of memory might need to be allocated for communication as the number of processors increases, spatial decomposition techniques allow the possibility of carrying out simulations involving very large numbers of atoms, where the same simulation would not fit into memory using replicated data models. Furthermore, the lower memory requirements of spatial domain decomposition techniques can have a significant impact in reducing execution times.

Of the codes available, NAMD [2] [5] and LAMMPS [6] claim to scale well in terms of speed-up with increasing numbers of processors, and the ability to deal with large numbers of atoms when a sufficiently large number of processors are available. These codes are freely available (and free) for academic research. NAMD is developed at the University of Illinois at Urbana-Champaign and LAMMPS development is concentrated at Sandia National Laboratories. The results presented in this report are produced from these two codes, although many of the algorithmic considerations will be equally applicable to any scalable molecular dynamics code.

3 Current Usage of Molecular Dynamics Codes

Although there is great scope for biological applications of Molecular Dynamics, where current research often uses packages such as Amber and CHARMM, other readily available codes such as DL_POLY allow the molecular dynamics technique to be used in widespread applications from chemistry to materials research.

The use of these established codes on HPC machines accounts for a large amount of total CPU time, but the lack of scalability of these codes limits their use to a small number of CPUs per

job as codes using the replicated data model are not able to utilise the full power of the resources available on large machines with hundreds or thousands of processors. Furthermore, this lack of scalability is an impediment when it comes to examining large simulations.

4 Scalable Code Features

The algorithms employed in the newer, more scalable molecular dynamics codes incorporate a combination of established methods and techniques, whilst where possible selecting those with the greatest computational efficiency.

Whilst reference has been made to the packages Amber and CHARMM, these same names are used to describe the force-fields which they employ in carrying out energy calculations. These force-fields are well established mathematical models which describe the interactions between atoms and include terms for electrostatics, van der Waals forces and bond interactions. LAMMPS and NAMD allow the input files to specify either Amber or CHARMM as the force-field to be used in calculations, thereby ensuring that users can have confidence in the methods used to perform simulations.

The local bond interactions are carried out at each time step using Verlet integration to advance the positions and velocities of atoms, but both NAMD and LAMMPS allow multiple timestep integration so that non-local electrostatic interactions (and possibly non bonded interactions) are calculated less frequently (up to 4 femtoseconds for electrostatics). In order to determine which short range forces are to be evaluated between atoms, the input must specify a cutoff distance, such that if two atoms are further apart than this distance then these forces are deemed to have a negligible value and are not calculated. The choice of this distance is a great factor in determining the amount of compuation which must be required since in a three dimensional problem such as a molecular dynamics simulation the amount of work grows as l^3, where l is the cutoff distance. Since the electrostatic interactions in a simulation are pairwise interactions between all atoms this part of the code is the most difficult to parallelise and is $O(N^2)$ in terms of the number of these interactions, where N is the number of atoms. For the electrostatic forces NAMD uses the Particle Mesh Ewald (PME) algorithm [7] which reduces the computational cost of full electrostatic evaluations to $O(N \log N)$.

Standard molecular and atomic coordinate input is provided using Protein Data Bank (PDB) files and X-PLOR input parameters together with a configuration file to specify the simulation to be performed. Both NAMD and LAMMPS are able to read Amber coordinate and velocity files, whilst NAMD is also able to read CHARMM and X-PLOR files as well as Gromacs files. This compatability could prove significant in persuading users to migrate to these scalable codes.

One very useful feature of NAMD for parallel simulations is the implementation of dynamic load balancing. The frequency with which load balancing is applied can be controlled by the user, but by default the simulation begins with an initial attempt at load balancing, then after 100 time steps the first dynamic load balancing is performed and then after every 4000 time steps. For simulations with large numbers of atoms, regular use of this feature can produce good parallel speed-up by ensuring a fairer distribution of work between processors.

5 Scalability of Amber

The following benchmark involved running a simulation which had been written as input to Amber and performing a translation of this simulation into NAMD format. The coordinate and trajectory

files were used as provided for Amber since NAMD is able to read these files, and the input parameters were then translated from Amber to NAMD. Results are shown in table 1.

Processors	NAMD Timing
1	30159
2	16096
4	8997
8	n/a
16	2637
32	1571
64	1209
128	1230
256	1199
512	1319

Table 1: Timings (seconds) for NAMD using Amber input for 27,109 atom simulation

Since the NAMD input was intended to mimic Amber as closely as possible, several features of NAMD which may decrease execution time were not used. For example, the simulation used full electrostatic evaluations at each step whereas NAMD allows the user to perform this part of the simulation less frequently (for example once every 4 femtoseconds). Modification of the NAMD input parameters to carry out PME every 4 femtoseconds showed a 20% to 30% reduction in execution times whilst maintaining the stability of the algorithm.

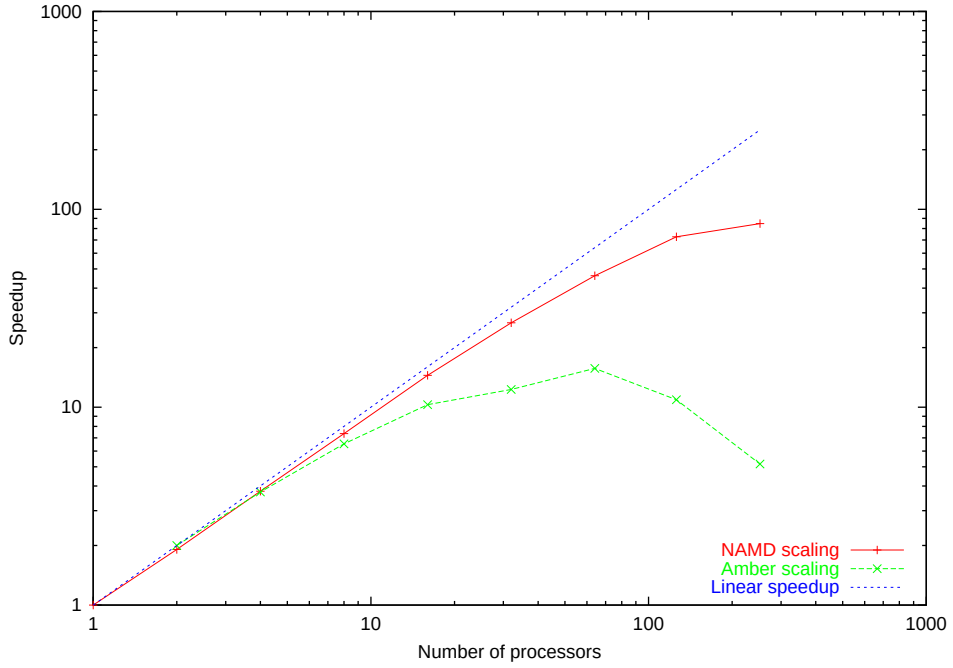

Figure 1: Scaling of NAMD and Amber

The timings in table 1 agree well with those contained on the NAMD web site [5] which demonstrate scaling as shown in figure 1. However, good scaling could be achieved by having a slower code, but figure 2 shows that for simulations involving more than 16 processors NAMD is able to outperform

Amber in terms of the time per step of the simulation.

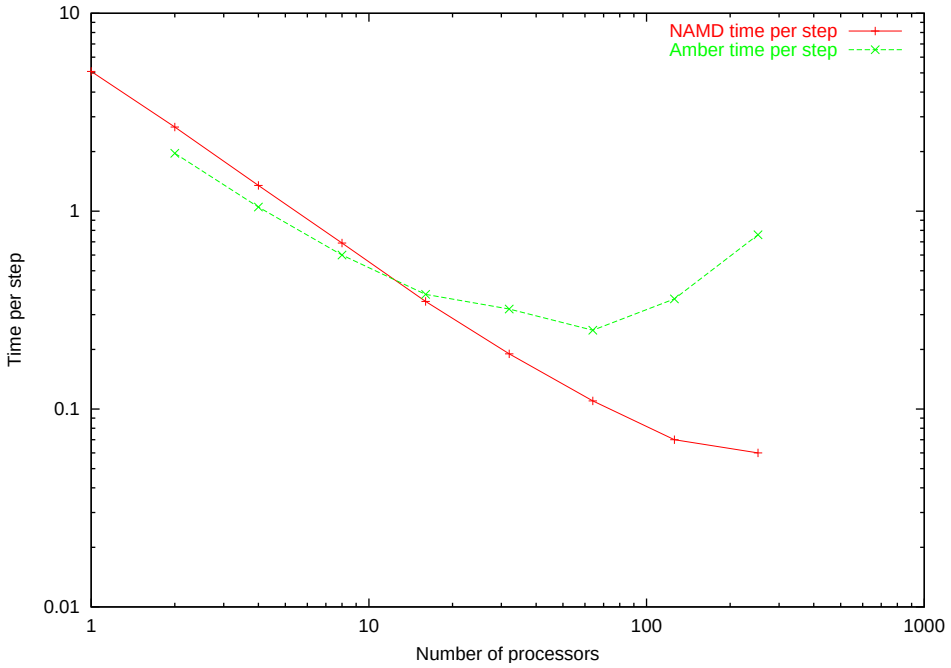

Figure 2: Time per step for NAMD and Amber

6 Benchmarks

Benchmarking was carried out on an SGI Origin 3800 with up to 512 MIPS 400 MHz R12000 processors available for parallel execution.

There were a variety benchmarks used to test the scalability of LAMMPS and NAMD, ranging from around 6000 to over 300,000 atoms. These benchmarks carried out 500 femtosecond simulations and therefore the serial I/O bound start-up time was a relatively large part of the benchmark time, particularly when using larger numbers of processors. Hence, speed-up using the time taken per step was decided to be the best guide to the scalability of the code for these problems.

The first set of simulations involved five problems ranging from 6028 atoms to 55909 atoms with a cutoff distance of 9 Angstroms using the LAMMPS package. The scaling is shown in figure 3.

These figures demonstrate that for real problems similar to the largest number of atoms which Amber can handle, the LAMMPS package demonstrates good scaling up to 128 processors. For the largest problem of 55909 atoms, the timings demonstrate some superlinear scaling up to 64 processors.

The same simulations (except for the 55909 atom example) were then carried out using the NAMD package, and the speedups are shown in figure 4. Whilst the speedups are not as good as those shown in figure 3 for LAMMPS, the actual timings are slightly faster.

The four simulations which were carried out in these examples were using the same time step for all force evaluations, however as has been pointed out, different forces can be carried out at

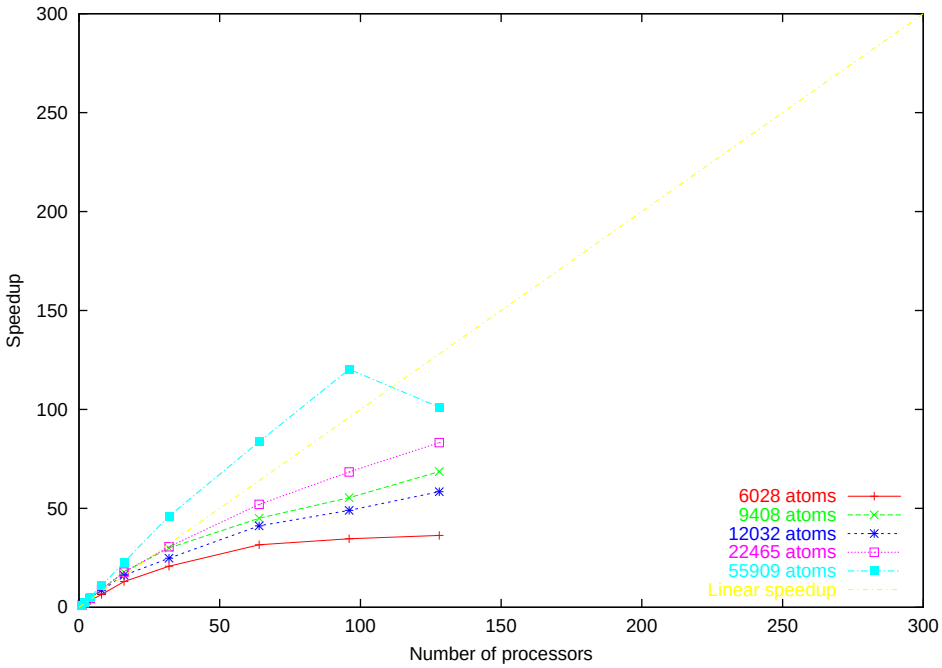

Figure 3: Scaling of LAMMPS for a variety of simulations

different intervals. Re-running the simulations but evaluating non-bond forces every 2 steps and full electrostatics every 4 steps produces the speedups shown in figure 5 which appears worse than with force evaluations at every step. However figure 6 shows the actual time per step for the full and multi-time stepping approaches for the 22465 atom simulation, and here the multiple time stepping approach is clearly beneficial.

These problems, whilst demonstrating the ability of NAMD and LAMMPS to compete with Amber, are too small to show the real scalability of these codes. For the true potential of these packages we need to analise larger problems.

The first of the larger simulations involves 92,224 atoms with PME evaluated every 4 femtoseconds. The results are shown in figure 7.

The results show scalability up to 500 processors on this system, with a speedup of 195 on 256 processors and a speedup of 332 on 500 processors. These speedups correspond well to the figures shown on the NAMD benchmarks web site [4].

An even larger simulation is of a large scale problem involving 327,506 atoms, again with PME evaluated every 4 femtoseconds. This simulation does around three quarters of the work per atom for the local electrostatics and van der Waals interactions compared to the previous benchmark, with a cutoff distance of 11 Angstroms compared to 12 (this corresponds to a volume ratio of the spheres of influence of approximately 0.77:1). The results are shown in figure 8.

Again, the figure shows, that for a larger problem, good speed-up (time per step) is attained for up to 384 processors (the maximum used).

The large time per step in this simulation provides for a good computation to communication ratio, and this problem should continue to scale well for a larger number of processors.

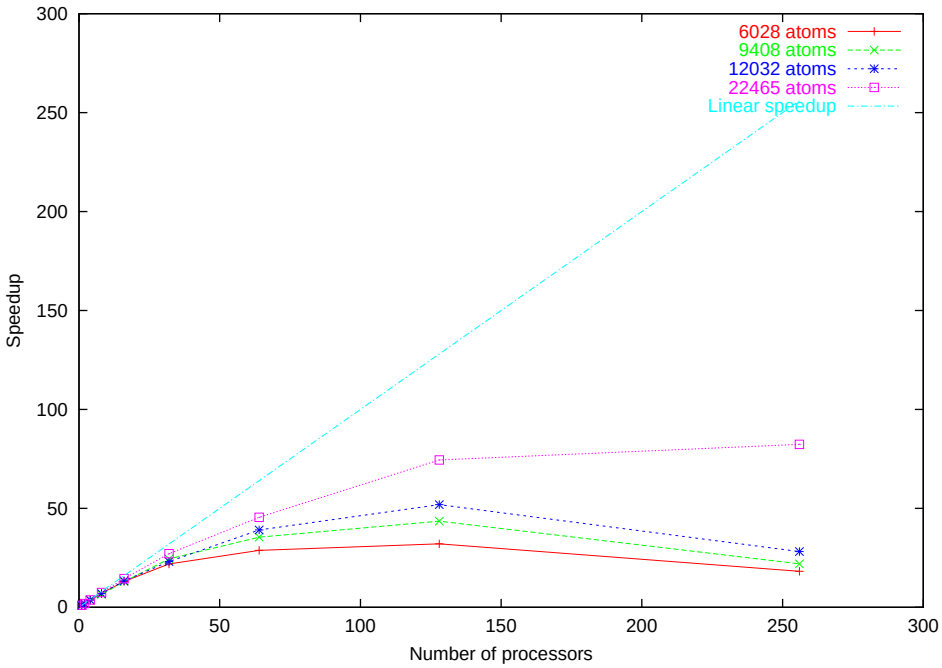

Figure 4: Scaling of NAMD for a variety of simulations

In terms of scalability of problem size, it should be noted that the NAMD code also scales approximately linearly with the number of atoms for a given number of processors (taking into account different PME constraints and the fact that the fourth simulation does less work per atom), and for larger problems, where PME evaluations are less frequent, we can expect good scaling in terms of both speed-up and the ability to cope with large problems.

In terms of the amount of time required to carry out these large calculations, a run of the 92,224 atom problem was able to carry out 710 picoseconds (0.71 of a nanosecond) of simulation using 256 processors in 24 hours, therefore a 1 nanosecond simulation should be possible in about a day and a half, bringing realistic large scale simulations within the grasp of researchers. Furthermore, at the end of this 24 hour period a good degree of convergence had been achieved.

7 Alternative Molecular Dynamics Codes

As already mentioned, there are a variety of codes/packages available for the solution of problems in molecular dynamics. NAMD and LAMMPS were selected for this evaluation, not just because they demonstrates good scaling, but also because input for the widely available Amber package can readily be used with an appropriate configuration file. However other codes are expected to appear including a new version of DL_POLY which incorporates a distributed data model, and should therefore offer good scaling to a large existing user base.

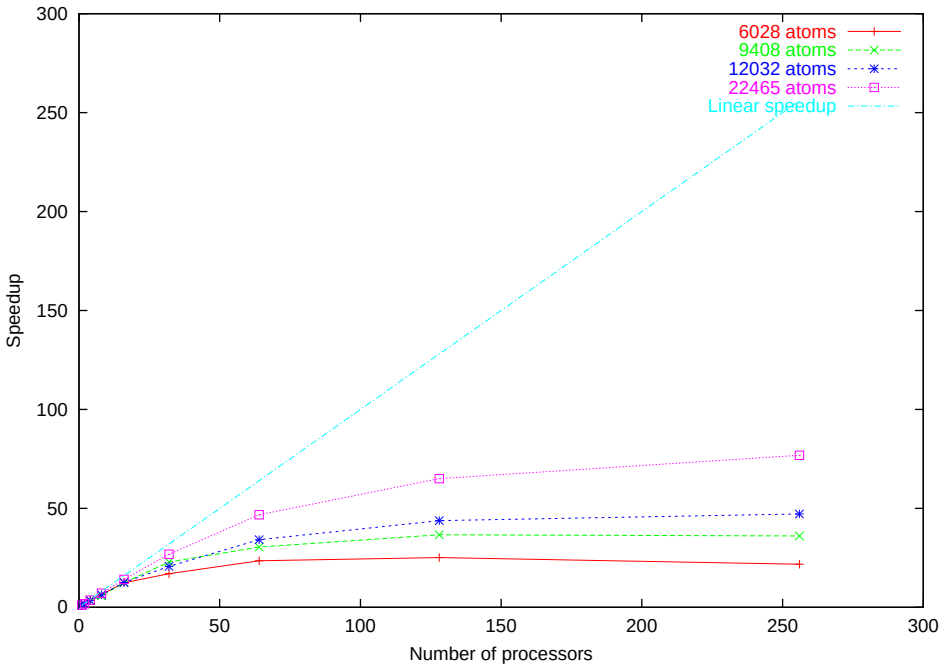

Figure 5: Scaling of NAMD using multiple time stepping for a variety of simulations

8 Conclusions

Scalable codes are now available for carrying out large molecular dynamics simulations on high performance computers and these codes scale well not only in terms of good speed-up, but also in their ability to handle large problems.

9 Acknowledgements

We would like to thank Jim Phillips at the University of Illinois for providing the input files to carry out the 90,000 and 300,000 atom benchmarks.

References

[1] Scuola Internazionale Superiore di Studi Avanzati. Mdbnch: A molecular dynamics benchmark. `http://www.fisica.uniud.it/ ercolessi/mdbnch.html`, cited May 2002.

[2] Laxmikant Kal, Robert Skeel, Milind Bhandarkar, Robert Brunner, Attila Gursoy, Neal Krawetz, James Phillips, Aritomo Shinozaki, Krishnan Varadarajan, and Klaus Schulten. Namd2: Greater scalability for parallel molecular dynamics. *Journal of Computational Physics*, 151:283–312, 1999.

[3] M.P.Allen, D.J.Tildesley. *Computer Simulation of Liquids*. Oxford University Press, 1987.

[4] Theoretical Biophysics Group of the University of Illinois. NAMD performance. `http://www.ks.uiuc.edu/Research/namd/performance.html`, cited May 2002.

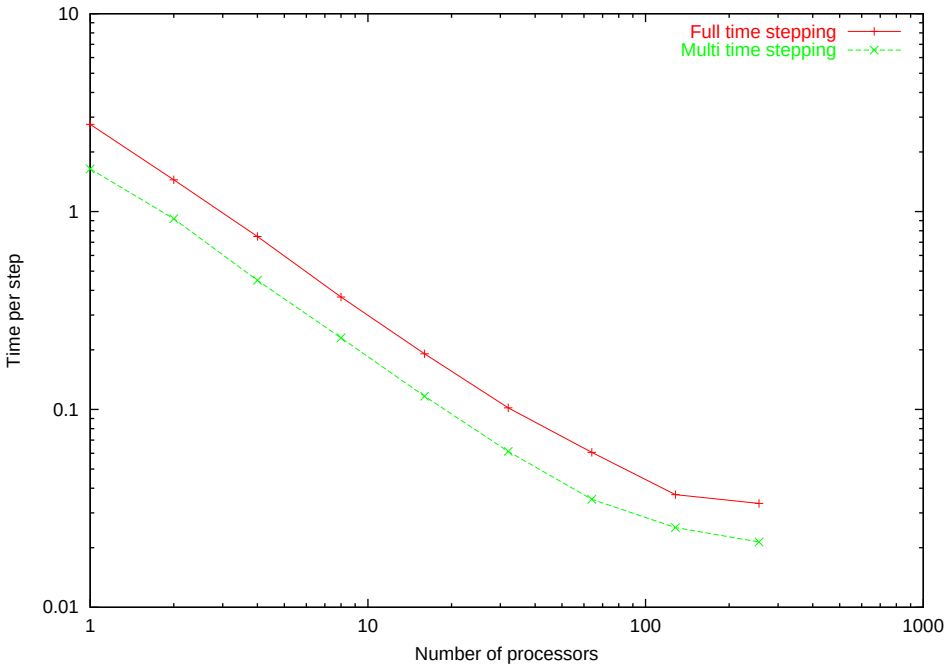

Figure 6: Time per step for 22465 atom simulation using full and multiple time stepping

[5] Theoretical Biophysics Group of the University of Illinois. NAMD web pages. `http://www.ks.uiuc.edu/Research/namd/`, cited May 2002.

[6] Sandia National Laboratories Steve Plimpton. LAMMPS web pages. `http://www.cs.sandia.gov/ sjplimp/lammps.html`, cited May 2002.

[7] T.A.Darden, D.M.York, L.G.Pedersen. Particle mesh ewald: An $N \log N$ method for ewald sums in large systems. *Journal of Chemical Physics*, 98:10098–10092, 1993.

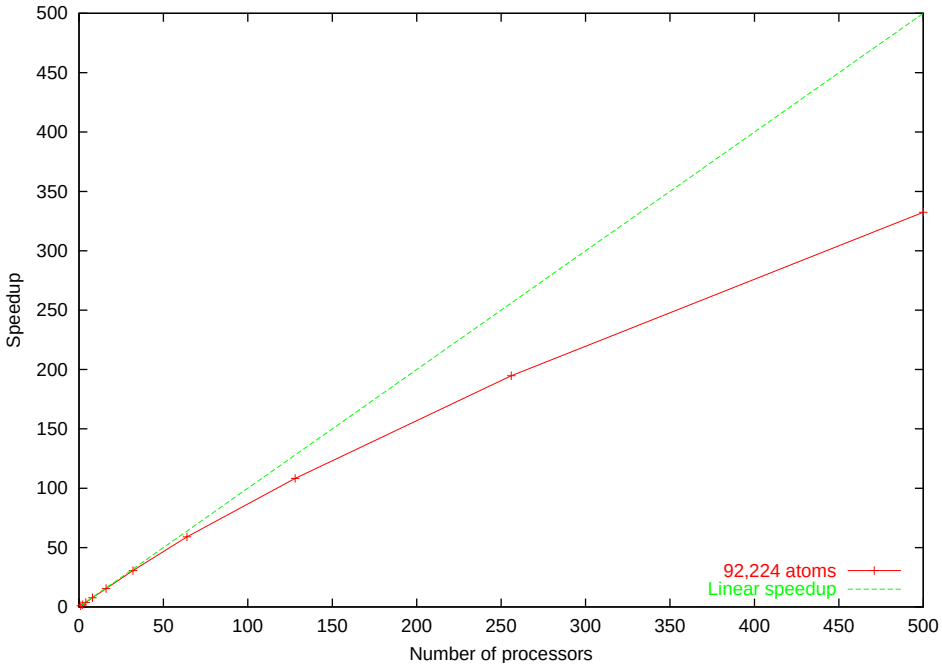

Figure 7: Scaling of NAMD for 92,224 atom simulation

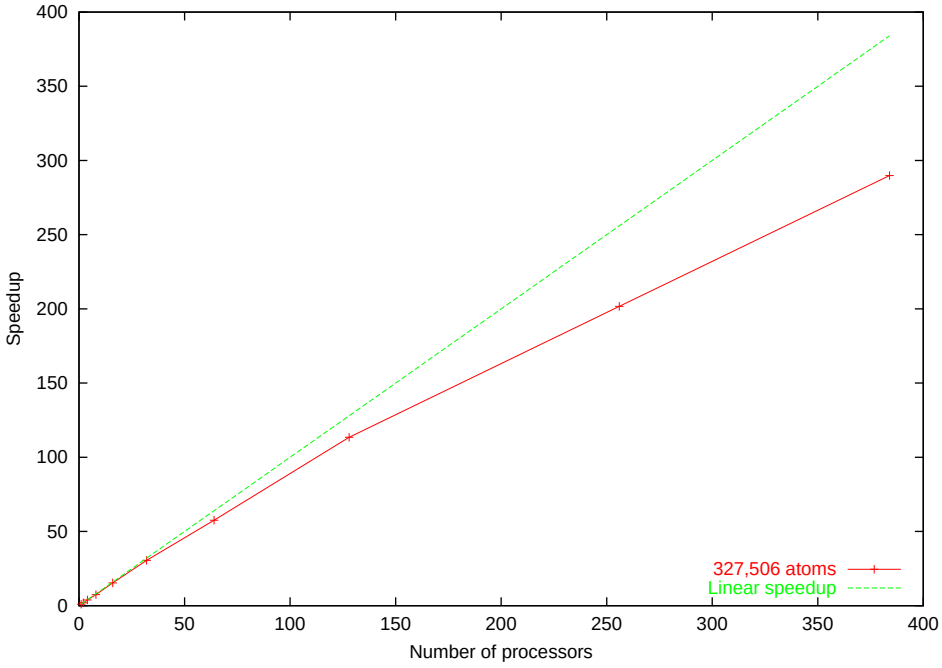

Figure 8: Scaling of NAMD for 327,506 atom simulation