
Received May 25, 1981.

The Relative Performances of Several Scientific Computers for a Liquid Molecular Dynamics Simulation

D. M. Ceperley
National Resource for Computation in Chemistry, Lawrence Berkeley Laboratory, Berkeley, CA 94720

In the last decade, the computer hardware of matter by simulations has become a very important area of theoretical chemistry. One goal of the simulations is to understand the properties of macroscopic systems starting from the Coulomb potential and Schrödinger equation. Although it is feasible that simulation methods can treat the complete many-body quantum problem, most simulations today assume a classical model with some effective interparticle potential. There are two common methods employed, Metropolis Monte Carlo (MC) and an effective algorithm used for calculating static properties of many-body systems. Molecular Dynamics (MD) is the term employed when Newton's equations of motion are solved to find equilibrium and non-equilibrium, dynamic and static properties of many-body systems. What all of these simulations methods have in common and their limitation is a processor fast enough to move hundreds of atoms, hundreds and thousands of times. What I want to discuss in this short note, are some of the computational characteristics of simulations and my experience in using a standard simulation program on several scientific computers.

While at the National Resource for Computation in Chemistry, I have developed a general classical simulation program, called, CLAMPS (for classical many particle simulator) capable of performing MC and MD simulations of arbitrary mixtures of single atoms. The potential energy of a configuration of N atoms at positions R = (r1, ..., rm) with chemical species (a1, ..., an) as is assumed to be a pairwise sum of spherically symmetric functions.

\[ U = \sum_{i<j} \phi_{a_i a_j} |r_i - r_j| \]

Where \( \phi_{a,b}(r) \) is the interaction between two atoms of type \( a \) and \( b \) and \( |r_i - r_j| \) means the minimum image distance consistent with periodic boundary conditions. In addition, there can be...
bonding potentials between certain pairs of atoms. If some of the atoms are charged there is another term in the potential energy arising from the interaction of a charge with the charges outside the simulation box: the Ewald image potential. (6) This can be conveniently written as

\[ U_c = \sum_k v_k |\rho_k|^2 \]  

(2)

Where \( v_k \) is a vector in the reciprocal lattice of the simulation cell, \( \rho_k \) is the Fourier transform of the charge density,

\[ \rho_k = \sum_i q_i \exp(i k \cdot r_i) \]  

(3)

\( q_i \) is the charge of particle \( i \), and \( v_k \) is the Fourier component of the long range potential. (6)

In simulations, computation of the potential energy and forces takes the vast majority of the computer time. The other operations, such as moving the particles, are usually much quicker. Shown in Table I is the FORTRAN coding needed to compute the pair sum in eq. (1). In a general purpose program such as CLAMPS one cannot assume that the pairwise interactions are simple enough to compute at each step, whereas, a table lookup is equally efficient for all systems. In CLAMPS the potentials and the derivative \(-r^{-1}\rho/dr\) are computed on a grid linear in \( r^2 \) at the beginning of the program and stored. Tables with the order of \( 10^4 \) entries usually give sufficient accuracy for most problems without any interpolation because of the statistical nature of the computation.

The coding in Table I illustrates the central problem of simulations. The number of pairs is \( N(N-1)/2 \). The number of floating point operations (FLOPS) per pair is about 25, assuming the branches are executed 50\% of the time. Thus for 100 atoms (a minimal simulation) we will need \( 1.2 \times 10^5 \) FLOPS for a single time step. The number of memory and indexing operations is similarly large. Typically one needs to execute between \( 10^3 \) and \( 10^5 \) time steps. Thus the simulations are limited by the number of floating point operations one can afford.

For systems which can be modeled with particles interacting with only short ranged forces (that is the potential can be neglected beyond several neighbor shells), the number of operations per time step will be proportional to the number of particles times the average number of neighbors of a given particle. For such models, simulations of \( 10^4 \) atoms are possible today on available mainframe as well as minicomputers. For many chemical systems, such as those containing macromolecules, one would like to work with still larger systems over many time steps. Even with today's computers, most chemical systems cannot be simulated without

Table I

<table>
<thead>
<tr>
<th>C LOOP OVER ALL PAIRS OF ATOMS I, J</th>
</tr>
</thead>
<tbody>
<tr>
<td>DO 1 I=1, NATOMS-1, J=I+1, NATOMS</td>
</tr>
<tr>
<td>C CALCULATE PERIODIC DISTANCES</td>
</tr>
<tr>
<td>R2=0.0</td>
</tr>
<tr>
<td>DO 3 L=1, NDIM</td>
</tr>
<tr>
<td>DX(L)=X(1,L)-X (J,L )</td>
</tr>
<tr>
<td>C ELL AND EL2 ARE THE BOX AND HALF THE BOX LENGTHS</td>
</tr>
<tr>
<td>IF (DX(L).GT.EL2(L)) DX(L)=DX(L)-EL2(L)</td>
</tr>
<tr>
<td>IF (DX(L).LT.-EL2(L)) DX(L)=DX(L)+EL2(L)</td>
</tr>
<tr>
<td>3 R2=R2+DX(L)**2</td>
</tr>
<tr>
<td>C IT AND JT ARE THE CHEMICAL TYPES</td>
</tr>
<tr>
<td>IT=ITYPE( I )</td>
</tr>
<tr>
<td>JT=ITYPE( J )</td>
</tr>
<tr>
<td>C CONVERT DISTANCE TO A TABLE ENTRY</td>
</tr>
<tr>
<td>LI=CSI ( IT, JT )-R2</td>
</tr>
<tr>
<td>C IF OUTSIDE TABLE POTENTIAL IS ZERO</td>
</tr>
<tr>
<td>IF(LI.GE.LMAX) GO TO 2</td>
</tr>
<tr>
<td>C LT IS THE TABLE FOR THIS INTERACTION</td>
</tr>
<tr>
<td>LT=LTABLE( IT, JT )</td>
</tr>
<tr>
<td>C LOOK UP POTENTIAL AND DERIVATIVE</td>
</tr>
<tr>
<td>V=V+EPS(IT, JT)*PTABLE(LI, LT)</td>
</tr>
<tr>
<td>FT=EPSF(IT, JT)*FTABLE(LI, LT)</td>
</tr>
<tr>
<td>C NOW ADD TO FORCES</td>
</tr>
<tr>
<td>DO 4 L=1, NDIM</td>
</tr>
<tr>
<td>F=F+DX(L)</td>
</tr>
<tr>
<td>FORCE( I, L )=FORCE(I, L)+F</td>
</tr>
<tr>
<td>4 FORCE( J, L )=FORCE(J, L)-F</td>
</tr>
<tr>
<td>2 CONTINUE</td>
</tr>
<tr>
<td>1 CONTINUE</td>
</tr>
</tbody>
</table>

FORTRAN Code for the pairwise sum of eq. (1).
making many simplifying assumptions. Both a supercomputer as well as better algorithms are necessary to tackle these problems.

For the purpose of comparing performance on different computers, I have used the Stillinger-Lemberg(7) model for water. This model contains central force interactions between charged oxygen and hydrogen atoms. The three different potential functions between O-O, O-H and H-H, are tailored to give the correct geometry and dipole moment for an isolated molecule and some of the pair bonding properties of two molecules.

Simulations of charged systems are very important. Common examples are plasmas, ionic solutions, dipole systems and electronic systems. Because all pairs are included in the sum of eq. (2), the computer time only depends on the number of atoms and the number of time steps. For this reason my results should be applicable to all similar systems. I will discuss here only results for molecular dynamics simulations. The situation for Monte Carlo is completely parallel, although the actual coding is different since atoms are moved singly rather than all together.

Because the atoms are charged, the Ewald image potential from eq. (2) must be used to account for the long-range Coulomb potential. In the following benchmarks, I have included all terms in the sum in eq. (2) for which $k < 6\pi/L$; this comprises 123 terms, and is adequate to represent the potential to one part in $10^6$. As long as the number of terms is held fixed, the computer time to evaluate eq. (2) will be proportional to the number of atoms while the pairwise sum in eq. (1) will take time proportional to the square of the number of atoms. Thus for large enough systems, it is the pairwise sum which dominates the calculation. The sines and cosines needed for $e_k$ are computed recursively. I will not discuss the computation of the Ewald sum in detail, because it is relatively specialized.

Computer Comparisons

In this section, I will discuss the programming considerations and timing results for the four computers on which I have tested CLAMPS. In all cases the code was not substantially changed. Essentially only the routines which performed the sums in eqs. (1) and (2) were modified. All changes were in FORTRAN or with FORTRAN callable routines. The timing results are not optimal, but rather typical of what could be achieved by a user in FORTRAN. The timing results given in Table II for systems containing 27 and 216 molecules (81 and 648 atoms). MFLOPS refers to the number of million floating point operations per second in executing the pairwise sum of Table I assuming each pass through consists of 25 floating point operations.

<table>
<thead>
<tr>
<th>Computer</th>
<th>81 Atoms</th>
<th>648 Atoms</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$T_p$</td>
<td>MFLOPS</td>
</tr>
<tr>
<td>VAX 11/70</td>
<td>0.35</td>
<td>0.23</td>
</tr>
<tr>
<td>CDC 7600</td>
<td>0.033</td>
<td>2.5</td>
</tr>
<tr>
<td>CRAY-1</td>
<td>0.0182</td>
<td>4.5</td>
</tr>
<tr>
<td>CRAY-1*</td>
<td>0.0070</td>
<td>11.6</td>
</tr>
<tr>
<td>VAX-FPSAP</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

DEC VAX 11/70

The VAX used, is located at NRCC in Berkeley, has a floating point accelerator, 2.5 M Bytes of memory, and was running version 1.3 of the operating system. The code was run in single precision (32 bits/word) and was found adequate to conserve energy and give satisfactory equilibrium properties. The code used to perform the pairwise sum is essentially that of Table I.

CDC 7600

The 7600 used is located at Lawrence Berkeley Laboratory, is approximately ten years old and has 65 K of 60 bit word fast memory (small core). Because CLAMPS has dynamic memory allocation, it is possible to fit a simulation in fast memory of up to about 2000 atoms as long as the potential tables are not too extensive. The compiler used was the standard CDC FORTRAN 4.8, OPT=2. The only difference between the CDC coding of the pairwise sum and that in Table I is that the periodic boundary conditions (loop 3) are handled by Boolean and shift operations instead of branches. Branches on the 7600 causes all parallel processing to halt.
The CRAY used is located at Lawrence Livermore Laboratory. Characteristics of the CRAY are described elsewhere in this volume. There is a large advantage in achieving vector rather than scalar code. This can be seen in Table II. Initially, the CDC version of CLAMPS was run on the CRAY with the time results showing it only slightly faster than the 7600. Several subroutines of CLAMPS were then vectorized and the simulation executed in approximately 1/5 the time. The vectorized version of the pairwise sum appears in Table III. The problems encountered in vectorizing this routine were:

1) The periodic boundary conditions in loop 3 contain 2 branches. Vectorization was achieved by using the FORTRAN callable vector merge function.

2) The branch for the case when the squared pair separation is outside the table will inhibit vectorization. The last element of the table has been changed to zero and all occurrence outside the table are truncated to LMAX. The rest of the code, which is not executed on the VAX or CDC 7600, is executed here. It is often necessary on a vector machine to increase the total number of floating point operations to achieve vector rather than scalar processing. The MFLOP rates reported here are computed on the basis of the original number of floating point operations. The extra ones added to achieve vectorization are not included.

3) The table look-ups for the force and potential can be done with the GATHER function. GATHER (N, A, B, INDEX) is equivalent to the FORTRAN statements.

\[
\text{DO 2 I = 1, N} \\
\quad A(I) = B(INDEX(I))
\]

Although GATHER is a scalar operation and rather slow, 12 machine cycles/element, (a machine cycle is 12.5 ns), GATHER is faster than computing all but the simplest inverse power potentials. By taking a square root takes 14 machine cycles/element and the entire pairwise sum takes an average of 98 machine cycles/pair. Note that temporaries are set up for the scaling factors of the potential, as well as the addresses for the start of the tables. These temporaries are changed only rarely and so do not affect the timing. They would be unnecessary if there were only one type of particle.

C LOOP OVER ALL PAIRS OF ATOMS I,J
  ITL=0
  DO 1 I=1,NATOMS-1
    ITL=ITL+1
    NC=NC-1
    C CHECK TO SEE IF WE NEED TO REFRESH OUR TEMPORARIES
    IF ( ITYPE(I).EQ.0) GO TO 20
    ITL=ITL+1
  C GATHER MAKES EPS(T,J)-EPS(TYPE(J),I) TYPE(I)
    CALL GATHER(NC,EPSF(I,I),EPS(I,ITL),ITYPE(I))
  C GATHER ALSO EPSF, CSI AND LTABLE
    CALL GATHER(NC,EPSF(I,I),EPSF(I,ITL),ITYPE(I))
    CALL GATHER(NC,CSI(I,I),CSI(I,ITL),ITYPE(I))
    CALL GATHER(NC,LTABLE(I,I),LTABLE(I,ITL),ITYPE(I))
  END DO 1
  C CALCULATE PERIODIC DISTANCES
  DO 4 L=1,NDIM
    T=SIGN(E(L),X(I,L))
  END DO 4
  C CVMDP(X,V,Z)=X IF Z GT.0 AND Y OTHERWISE
  DO 6 J=1,NATOMS
    DX(J,L)=CVMDP((X(I,L)-X(J,L))-T,X(I,L)-X(J,L))
  END DO 6
  C MAKE R2 INTO TABLE ENTRIES WITH LMAX BEING MAXIMUM
  DO 8 J=1,NATOMS
    INDEX(J)=LTABLE(J)+MINO(LMAX,INT(CSIT(J)+R2(J))
  C GATHER V(R,J) INTO A VECTOR
    CALL GATHER(NC,R2(INDEX(J)),PTable,INDEX(I))
  C SUM THEM UP
    VTotal=VTotal+SDOT(NC,R2(INDEX(J)),1,EPST(I,J))
  C GATHER DERIVATIVES FROM PTABLE INTO R2
    CALL GATHER(NC,R2(INDEX(J)),FTable,INDEX(I))
  C MULTIPLY BY DISPLACEMENTS AND ADD INTO FORCE VECTORS
    DO 8 L=1,NDIM
      FORCE(I,L)=FORCE(I,L)+SDOT(NC,R2(INDEX(J)),1,DX(I,L),1)
  END DO 8
  C CONTINUE

FORTRAN code, optimized for the CRAY, which performs the pairwise sum of eq. (1).
9. Ceperley  Liquid Molecular Dynamics Simulation

4) As we have seen on the CRAY the ability to gather data together is essential. Memory speed must be commensurate with floating point speed. When nearest neighbor tables are used fast scatter operations are also needed. The two essential random memory operations needed are:

\[ B(I) = A(INDEX(I)) \]
\[ B(INDEX(I)) = B(INDEX(I)) + A(I) \]

5) Except for the above gather-scatter operations, simulations are easily vectorizable as defined by the CRAY FORTRAN or the CYBER 200 FORTRAN. Typical vectors have 50 to 500 elements each.

6) The Basic Linear Algebra Subroutines (BLAS) are a convenient way of maintaining efficiency and portability. They should be extended to include such things as GATHER, and vectorized EXP, SQRT, SIN and COS.

Acknowledgment

This research was supported in part by the Office of Basic Energy of Science of the U. S. Department of Energy under Contract No. W-7405-ENG-48 and by the National Science Foundation under Grant No. CHE 7721305. Special thanks to Dr. M. Rao and Prof. B. Berne for making available and assisting me in the use of the computing facilities in the Chemistry Department at Columbia University.

Literature Cited


5. Contact the Quantum Chemistry Program Exchange, Department of Chemistry, Room 204, Indiana University, Bloomington, Indiana, 47405 for CLAMPS.


The Need for Supercomputers in Time-Dependent Polymer Simulations

The development of polymer science has had a profound effect on technology. Polymer materials have important applications because of their mechanical properties, dielectric, flow, and thermal properties which are tightly controlled by the chemical structures of the monomers. Clearly, the smallest cross-sectional area of a fundamental particle of polymer configurations and the mechanism by which the polymer chain moves from one configuration to another is a function of the chemical structure of the monomer. A number of theoretical models have been developed. In each model, the polymer chain was represented by a random walk between a set of points in space. The points represent the polymer atoms and the steps allowed. The simplification of many polymer properties, however, makes the model too complicated for computer simulation. Computer methods have been employed to examine model polymer properties. For example, Monte Carlo methods have been used to calculate hard potentials on all-polymer systems. Many models are being developed to examine the effects of excluded volume on polymer properties. In this manner, it has been possible to find computational simplicity and the dependence of the average mean square radius of gyration, $S^2$, on the number of units in a polymer chain $N$ for a chain with excluded volume. These results hold for infinitely dilute systems, i.e., one chain.


The BLAS (Basic Linear Algebra Subprograms) are a collection of 38 FORTRAN callable routines that perform many of the basic operations of linear algebra. Contact International Mathematical and Statistical Libraries, IMSL, for more information.