The Evolution of Molecular Dynamics Codes From PVP to MPP

Barry C. Bolding, Cray Research Inc., Eagan, MN

ABSTRACT:*The current push to place computational chemistry application on MPP machines is driven by the bright prospects for scalability that applications such as molecular dynamics provide researchers. By using CHARMM as a prototype I will discuss the evolution of molecular dynamics applications from the traditional PVP to MPP environments. The positive and negative aspects of such simulations on each platform and will be presented. I will also discuss future trends in molecular dynamics for the Cray-T3D/T3E and PVP platforms.*

Introduction

Molecular Dynamics (MD) is the modelling of the behavior of atoms and molecules under the influence of the interactions between them. The commercial, industrial and academic applications of such simulations abound in the fields of drug, materials and polymer design. In recent years the desire to understand the behavior of larger and more complex systems has driven research in how to develop more efficient computational methods. The high peak performance promises of massively parallel computers (MPPs) and the intrinsically parallel nature of the molecular dynamics problem is pushing this research to the forefront of algorithm development for MPPs. If we wish to take advantage of this performance advantage on MPPs then we face the difficult question of whether to take existing powerful MD codes (such as CHARMM, AMBER or DISCOVER) and port them to the MPP platforms wholesale, or to develop new codes which are designed from the ground up to take advantage of parallelism. In this paper I will discuss some of the issues involved in this problem.

The Molecular Dynamics Algorithm on PVP's and MPP's

In order to compare and contrast the PVP and MPP approaches to MD, let us first consider algorithms that have been in use for years on both vector and scalar machines for performing parallel shared memory MD. In molecular dynamics each atom interacts with atoms in its nearby region of space. If a list of these "neighbors" is kept for each atom, then it is fairly easy to parallelize by distributing the atoms and their associated list to different processors [1].

```
do k=1, NCPUS  ! distribute over processors
    do i=1, Natoms, NCPUS
        do j=1, Neighbors(i)
            calculate interactions and update
            interactions on i
        enddo
    enddo
    ! update interactions which may conflict
    cdir$guard
    update interactions on j
    cdir$endguard
enddo
```

Notice that the stride of NCPUS on the atom loop above results in what is termed an atom-decomposition.

For MPP platforms the MD algorithm take the following general form.

1. Decompose System across processors
2. Calculate local interactions on each node
3. Communicate interactions which cross boundaries between nodes.
4. Move particles
5. Go To (2) or (1) as needed

The decomposition of the system across processors takes on three common forms; atom decomposition (AD), force decomposition (FD) and spatial decomposition (SD). The relative merits of these three methods for short-ranged interactions are detailed by S. Plimpton [2]. As already mentioned above, the PVP algorithm is an AD method. This is usually the first method used when porting an MD code to the Cray-T3D, but is the least scalable for large numbers of processors.

!decompose system

```
calculate locally A_first(mype) and A_last(mype)
do i = A_first(mype), A_last(mype)
    do j=1, Neighbors(i)
        calculate interactions and update local
        interactions of i and j
    enddo
enddo
global sum of interactions on i and j
```

One can see that this algorithm doesn't usually involve a large restructuring of data, but simply defining some new local pointers which keep track of the molecules on each PE. The AD method is currently implemented in CHARMM and AMBER for the T3D.

The spatial decomposition typically exhibit more optimal parallel performance for very large, somewhat homogenous systems. Unfortunately it is somewhat more difficult to implement, needing some rearrangement of data structures in order to achieve excellent performance. Some of the advantages of a spatial decomposition include, more efficient creation of neighbor lists and compatibility with algorithms for determining long range interactions (Ewald[3] or Cell Multipole Methods[4]). Figure 1 depicts a cell decomposition for a biochemical system. Each processor is given a cell or set of cells to work with in the MD simulation. Communication is primarily need to communicate between cells for those interactions which cross cell boundaries.

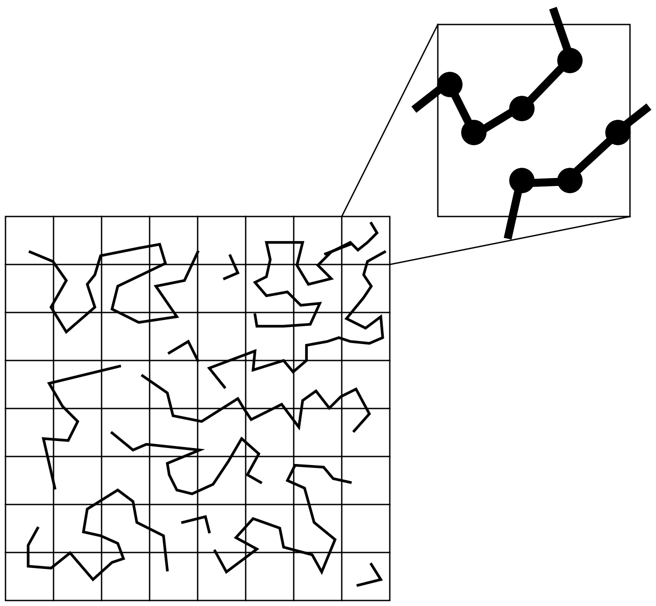

Figure 1: Spatial Decomposition

Molecular Dynamics: Why and When is it Scalable.

For PVP and scalar parallel architectures one is used to talking about the parallelism in a code. If a code can utilize 2 to several dozen processors efficiently then we can term the code parallel. To differentiate, if a code can utilize several hundred processors efficiently, then we will call it scalable. Scalability in MD arises from two distinct parts, size of a system and the complexity of the interactions. With simple two particle Lennard Jones interactions

$$V(r) = \frac{a}{r^{12}} - \frac{b}{r^6}$$

the amount of work done in calculating the interaction between two atom is quite small (several dozen flops). To insure that communication does not become a bottleneck then each processor needs to be working on several thousand of the these types of interactions.

If the complexity of the interactions are increased then the communication can be negligible for even a small number of interactions. Of course in the limit of few interactions per-processors the load balancing may also be an important issue. For biochemical and polymer MD the interactions are often quite complex.

$$V(r, \theta, \phi, q_i) = V(r) + V(\theta) + V(\phi) + V_q(r))$$

When the interactions between atoms depend on several complex functions of distance, angle, and charges between atoms. This type of interaction might involve calculating several hundred to several thousand flops per interaction. An extreme situation for this type of complexity occurs in Density-Functional-Theory Molecular Dynamics (DFT-MD). In this case the interactions are so complex that millions of operations are performed per-atom in calculating the interactions in a very small system.

In addition to the complexity of the system and its effect on the scaling, There are scaling laws that govern the behavior of the differing decompositions in molecular dynamics. If N is the number of atoms in a system and P is the number of processors available, it has been shown for short ranged interactions that the scaling for large numbers of processors is optimal for a spatial decomposition algorithm. Table 1 shows these scaling laws.

TABLE 1.

Algorithm	**Scaling**
Atom Decomposition	O(N)
Spatial Decomposition	O(N/P)

CHARMM as a prototype

CHARMM is a molecular dynamics code developed by several hundred scientists and programmers at Harvard and throughout the world over the past 20 years. It contains roughly 200,000 lines of Fortran and C code. The code has a tremendous amount of functionality built in over the years. The goal of several groups has been to port these functionalities in a orderly fashion to parallel platforms. Unfortunately many of the data structures and algorithms were developed prior to the advent of parallelism. This creates a tremendous inertia to change within the code. Great progress has now been made on CHARMM and several of the major types of simulations now run efficiently on the Cray-T3D. An example of the parallel performance can be seen for the myoglobin benchmark developed by Dr. B. Brooks (see Table 2).

TABLE 2. Myoglobin Benchmark with CHARMM

NCPUS	T3D Time(sec)	Speedup	C90 Time(sec)
8	5505	7.8	265
16	2765	15	
32	1439	31	
64	754	54	
128	454	91	
256	309	168	

The parallelism for this simulation is quite good. Unfortunately for 256 processors the efficiency is already at only 67%. This benchmark is probably achieved close to the current optimal scalability for CHARMM. The myoglobin benchmark doesn't use periodic-boundary conditions(PBC). If PBC's were used, the efficiency would be decreased. This indicates that additional code work will need to improve the parallelism of the PBC code.

Other Approaches

Several groups around the world are taking different approaches to building efficient parallel MD codes. One of these approaches is to build an entirely new MD codes from scratch, beginning with data structures and algorithms that are intrinsically designed for scalability. This approach is exemplified by the work being done on DL_POLY the CCP5 research group at Daresbury[5] and by K. Schulten and co-workers[6]. Cray Research Inc. is also actively involved in developing such a scalable MD code by working with a group of scientists at Bristol-Myers Squibb, Dupont, Lawrence Livermore and Sandia National Labs. These investigators are involved in 2 CRADAs (cooperative research and development agreements) which have the goal of producing a massively parallel molecular dynamics code for use with large-scale molecular systems.

The idea behind all of these development groups is that if one starts with a code that is efficiently scalable then one can add functionality in a systematic fashion that maintains the scalability. Most the results from these various groups are still in the preliminary stages, but they all have in common the desire to implement long-ranged coulombic interactions, efficiently scalable decompositions, and highly portable code. In addition, the CRADAs that Cray is involved with are implementing full molecular force-fields for simulating biomolecular systems. This work is being spearheaded by Steve Plimpton at Sandia National Labs.

Conclusions

We see that it can be straightforward to utilize a molecular dynamics code like CHARMM as an efficient parallel tool for reasonable numbers of processors by utilizing existing atom decompositions. The question of scalability of such codes to hundreds of processors remains open. Perhaps in optimal circumstances one can obtain reasonable efficiencies out to 200-300 processors. For larger numbers of processors several groups of researchers are hoping to develop codes which will allow simulations of molecular systems which have been heretofore out of reach. We eagerly await the results of these differing approaches and the scientific knowledge that will be derived from them.

References

[1] J. E. Mertz, D. J. Tobias, C. L. Brooks III and U. C. Singh, J. Comp. Chem., **12**, 1270, 1991

[2] S. Plimpton, J. Comp. Phys. **117,** 1, 1995

[3] R. K. Kalia, S. de Leeuw, A. Nakano and P. Vashishta, Comp. Phys. Comm. **74,** 316, 1993

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

[5] W. Smith and T. Forrester, Private Communication, via http://www.dl.ac.uk/CCP/CCP5/main.html

[6] J. A. Board Jr., L. V. Kale, K. Schulten, R. D. Skeel and T. Schlick, Comp. Biology, Winter, 19,1994