Particle Simulations on Massively Parallel Machines
Steve Plimpton
Sandia National Laboratories

Abstract

A wide variety of physical phenomena can be modeled with particles. Such simulations pose interesting challenges for parallel machines since the computations are often difficult to load-balance and can require irregular communication. We discuss the size of problems that can be simulated today, obstacles to higher performance, and areas where algorithmic improvements are needed. The relevant issues are illustrated with two prototypical simulations: a Monte Carlo model of low-density fluid flow and molecular dynamics.

Introduction

Particle simulations come in two flavors, depending on whether the force model for particle interactions is short-range or long-range. In the short-range case, each particle interacts only with nearby particles. Examples include fluid flows dominated by collisions, molecular dynamics simulations of solids and liquids, and particle-in-cell (PIC) simulations which track propagating electromagnetic waves. The computation in short-range simulations scales linearly with N, the number of particles. Thus on a parallel machine with P processors the run time will scale optimally as N/P. By contrast, particle simulations with long-range forces must account for interactions between all N^2 pairs of particles. Simulations with slowly varying potentials due to Coulombic and gravitational forces are examples in this class. Algorithms that are hierarchical in nature or employ multipole expansions are typically used for large systems to avoid performing N^2 work.

In this paper, only short-range force particle simulations are considered. Achieving optimal N/P scaling is a challenge on a large parallel machine because it requires fully exploiting the locality of the forces. This is often done by superposing a grid on the simulation domain to assist in locating nearby particles. In a few hours or days of CPU time, current-generation supercomputers can perform million-particle simulations for tens-of-thousands of timesteps or many-thousand-particle simulations for millions of timesteps. It is important to note that there are interesting scientific problems in both domains. In other words, as computers become more powerful, researchers will want to perform simulations with more particles AND simulations for longer timescales. The computational workload in both cases is roughly equal due to the O(N) scaling of the problem. However, long timescale simulations of moderate numbers of particles can be harder to parallelize effectively for thousands of processors. In the next two sections, examples of particle simulations needing both more particles and longer timescales are discussed.

Monte Carlo Fluid Flow

Direct Simulation Monte Carlo (DSMC) is a simulation model of low-density fluid flow where each particle represents roughly 10^{12} physical molecules of a particular chemical species. Particles flow through the simulation domain interacting with each other as well as boundaries and surfaces. At each timestep all the particles move independently (without collisions). At the end of the
DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.
timestep a grid is used to locate particles near each other. All the particles in a particular grid cell then perform random collisions and chemistry reactions with each other via Monte Carlo rules, resulting in a new velocity distribution. By averaging the particle and collision statistics over time, quantities such as density and temperature can be profiled throughout the flow and on surfaces in the simulation domain. The DSMC model is the method of choice for simulating flows where the continuum Navier-Stokes equations break down due to low densities. Equivalently, these are flows with a high Knudsen number, which is the ratio of the mean-free-path molecules travel between collisions to the characteristic length scale of the flow geometry. Upper atmosphere flows around space vehicles and flows in high-vacuum semiconductor processing chambers are examples of commonly simulated problems using the DSMC technique.

Computationally, there are many trade-offs to be considered when developing a DSMC code. By making grids irregular so that small grid cells can be placed in higher-density regions of the flow, weighting particles to increase counting statistics in low-density regions, and advancing the timesteps more rapidly in some parts of the simulation domain, accurate answers can be arrived at with orders of magnitude less computational work. However, these features add coding complexity and make the DSMC simulation more difficult to parallelize. They also tend to produce a code that does not vectorize well. Developing a DSMC code for our nCUBE 2, a MIMD parallel machine with serial processors, we decided including these complexities made good sense. The resulting code has allowed us to perform several-million particle simulations for tens-of-thousands of timesteps and track very large density variations (6 orders of magnitude) in simulated flows around the NASA wake shield. The code achieves a 30-50 times speed-up over a single Cray Y-MP processor on 1024 nCUBE processors for large problems. However, the reason for the dramatic speed-up over the Cray is that the code is essentially serial. For the next-generation of processors such as the i860 and CM-5 vector chips, our current code will not achieve high single-node performance rates. Reducing the complexity of the simulation to enable more vectorization may be necessary to simulate larger particle ensembles.

Large density fluctuations and rapid movement of particles on the grid also pose a challenge for parallel implementation of DSMC models. Even with variable-sized grid cells there can be 100 times more work to do (per timestep) in cells with many particles than in sparsely populated cells. We have found a scattered decomposition of cells to processors works best so that on average each processor owns some costly and some cheap cells. However, this maximizes interprocessor communication since large numbers of particles move to new processors at every timestep. We estimate a factor of 2-3 speed-up could still be gained in our DSMC model by minimizing communication while maintaining load-balance. This would require some kind of clustered decomposition of the irregular grid which could be adjusted rapidly enough to track the rapid density variations in the transient regime of the flow. We have experimented with fast load-balancers such as inertial methods, but have not yet found a completely satisfactory answer. In contrast to load-balancers for PDE grids where the "optimal" decomposition is important enough to justify a high cost in computing it, our judgment is that particle simulations such as DSMC are better served by fast, sloppy balancers. Speed is more important than accuracy, since the local particle densities change so quickly. A related issue is that the balancer should produce a new decomposition that requires minimal data movement; its cost may be prohibitive if it requires millions of particles to be moved to new processors.

Finally, the issue of adaptive gridding is one that is largely unexplored for DSMC simulations. As
in other fluids or particle simulations, refining and coarsening grids appropriately as the transient flow proceeds to steady-state can offer a large computational savings. For parallel implementation, any adaptive gridding scheme must be coupled to the load-balance issues discussed in the previous paragraph.

Molecular Dynamics

We now turn to a particle simulation where many interesting scientific problems can be addressed with modest (tens-of-thousands) numbers of particles but which require very long timescales to simulate. This is the atomistic simulation method known as molecular dynamics (MD). It is an empirical model of atomic and molecular motion. All quantum mechanical and electronic effects are bundled into energy functionals that describe the forces acting on each atom. In organic systems such as polymers and proteins there are two kinds of forces: non-bonded forces which are pairwise interactions of each atom with its neighbors and bonded forces which are 2-, 3-, and 4-body interactions within the topology of a molecule resulting from covalent bonding. The computational task in MD is to compute the forces on each atom (which can be done in parallel) and to integrate the equations of motion for the system. Conformational and transport properties of molecules can then be deduced. However, because the femtosecond timescale for atomic vibrations is so short, many millions of timesteps must be simulated to account for even nanoseconds-to-microseconds of real time. Since many interesting physical events such as protein folding, catalytic reactions, and diffusion of polymers in zeolites, occur on the microsecond-to-millisecond timescale, it is hoped that faster parallel machines will enable longer simulations to be done, not just larger ones.

The methods commonly used for parallelizing MD simulations include atom-decomposition and spatial-decomposition techniques. In the former, each processor owns \( N/P \) atoms for the duration of the simulation. It computes forces on those atoms and updates their positions. The updated information must be shared with all \( P \) processors, which requires all-to-all communication that scales as \( N \), independent of \( P \). Thus this method scales poorly as the number of processors is increased. By contrast, spatial-decomposition methods partition the simulation domain into \( P \) pieces, one per processor. Each processor computes forces on and updates the positions of only the atoms in its sub-domain. For short-range forces this only requires communication with nearby processors to acquire the needed information. Thus in the large \( N \) limit, the method scales optimally as \( N/P \). However, for organic systems, spatial-decomposition methods are difficult to implement efficiently. First, the atoms in organic systems are often non-uniformly dense. Examples include free surfaces in an organic thin film or a globular protein filling a spherical volume. In either case it is difficult to break the domain into \( P \) equal-sized, simply-connected pieces so that each processor has equal work to do and can perform minimal, regular communication. Second, organic systems have strong Coulombic interactions. While these are often truncated to save computational work, the cut-off lengths can be quite long. Simulating tens-of-thousands of atoms on a thousand processors means the sub-domain owned by a processor is small compared to the cut-off and considerable communication is needed to acquire nearby atom positions. Finally, computing the 2-, 3-, and 4-body forces in organic systems requires topological information as well as atom positions. Passing this information from processor to processor as molecules move through the simulation domain requires complex coding and extra communication, introducing more parallel inefficiency.
Because of these drawbacks, we know of no general MD simulation for organic systems (e.g. CHARMM, GROMOS) that has been implemented in parallel on hundreds-to-thousands of processor systems using a spatial-decomposition method. Such an implementation is clearly needed for very large N simulations and might take advantage of advances in irregular and/or adaptive gridding techniques discussed in the previous section. For moderate-sized systems we have developed a new parallel MD algorithm we call force-decomposition which has several advantages for organic system simulation. It treats the set of non-bonded pairwise interactions as a NxN sparse matrix, partitioned into P blocks, one per processor. With this partitioning all communication in the algorithm can be performed between small groups of \( \sqrt{P} \) processors, using vectors of length N/\( \sqrt{P} \). This is in contrast to the atom-decomposition method which must perform true all-to-all communication on vectors of length N among all P processors. Thus a factor of \( \sqrt{P} \) is gained in communication and memory cost with this algorithm. In practice this allows many more processors to be used effectively. We have seen speed-ups of 2-3x in our organic MD codes on 512-1024 processors of the Intel Delta or nCUBE 2 using a force-decomposition method over an atom-decomposition method. The key advantages of the new method over spatial-decomposition approaches are that it preserves the simplicity and geometry-independence of the atom-decomposition method. Thus it is efficient for irregular and long cut-off organic systems where the spatial methods have difficulty. Its N/\( \sqrt{P} \) scaling is certainly not optimal for large P but in practice it has turned out to be a reasonably fast method for many of the moderate-sized systems we are interested in studying.

Finally, unlike the DSMC simulations discussed in the previous section, the kernel operations in MD simulations are readily vectorizable. Since the algorithms discussed above are already very parallel, we anticipate most of the advances in speed on the coming generation of parallel computers such as the Intel Paragon and CM-5 will come from boosting single-node performance. The challenge will be to structure memory usage and pipeline operations for optimal performance on chips like the i860 and CM-5 vector units.

**Summary**

While particle simulations are inherently parallel, many interesting obstacles and issues must still be addressed to achieve high performance on massively parallel computers. These include (1) parallel methods for systems with moderate-sized N to allow longer timescales to be simulated, (2) boosting single-node performance by more careful attention to data movement and vectorizable operations, (3) dynamic load-balancing methods which can quickly track particle density fluctuations, and (4) adaptive gridding techniques which can track the density variations to both minimize computational effort and maximize parallel performance by putting processors to work in the most critical portions of the simulation domain.

**Acknowledgments**

Much of the work described here has been collaborative efforts with other Sandians. In particular, I thank Tim Bartel, Grant Heffelfinger, and Bruce Hendrickson. Reprints describing the Monte Carlo and molecular dynamics work in more detail are available from the author at sjplimp@cs.sandia.gov.
END

DATE FILMED

8/19/93