Parallelizing a Direct Simulation Monte Carlo Method for Micromechanical Systems

Rajeev Surati
*Address: Room 439, 545 Technology Square, Cambridge MA 02139.
Web: http://www-swiss.ai.mit.edu/ raj
raj@ai.mit.ed

1 The Application

Several researchers are interested in fabricating tiny micro-machines. A miniature gas powered engine, is one example of a machine they would like to make. It might be used to replace a battery to powerother small systems. One problem these researchers face is they need to test and validate their designs. People have not developed a good intuition for fluid flow at this scale. For the fluids of interest (gases), the continuum assumption which allows you to utilize normal Navier-Stokes equation is not valid and thus will not explain observable phenomenon. A measure of the validity of this assumption is the Knudsen number. This is a measure of the mean free-path between particles and the feature size of the system. For air at standard conditions this mean free path is 60 nm. For micromechanical structures, small defects and features of boundaries and structures cause this number to become greater than one and thus make the continuum assumption in correct. This means you can not simply aggregate particle behavior together and treat them as a density. A molecular method is at least required to simulate these systems. One such scheme that approximates the flow is known as Direct Simulation Monte Carlo. The purpose behind parallelizing this code is to allow the simulator to be able to simulate much larger structures than it could on a single processor. On a single processor the times are bearable for certain size systems, but for more interesting problems a parallel system is necessary.

A serialized version was implemented by Ed Piekos in Course 16 for his SM thesis. I implemented a parallel version on an HP cluster as well as an IBM SP2.

At a common pressure there are about 10^{19} particles in a cubic centimeter, if we kept track of every single particle's position, velocity, and internal energy, a formidable amount of memory would be required to accomplish this simulation. One assumption made is that you can aggregate some of the particles together as a unit. Thus one represents several points with one mass(A quasi-continuum). Another, is that in order to really do this simulation with n particles one would need to do n^2 interactions to see what collides with what. For a million particle elements thats way too many interactions to do feasibly. Multipole could take care of all central field interactions, but that is not the overriding force here. Instead, what one does in this method is to step the system for a small portion of the time it takes a particle to travel the mean free path distancce ignoring collisions and then colliding the particles so that certain statistical properties of the system are satisfied. The geometry is triangulated into cells. Each cell has some average number of collisions in it. The cells are on the order of the mean free path length. This turns the n^2 calculation into effectively an order order n one. Also,

*This project was done with the support of IBM with their SP2 installation on campus and was done under the auspices of Department of Energy Computational Science fellowship with Gerry Sussman and Kenny Breur

boundary conditions such as particle flow into the system or rigid walls are also easily accomplished in the simulation.

2 Parallelization Strategy

I chose to utilize message passing as implemented with the MPI standard. Particle elements are the basic unit in the system. They have a processor associated with them at any particular time and they can migrate from one cell to another. The user inputs a structure which gets triangulated by the system. Then the system is partitioned. Right now this is done by cutting the system up into geometrically separated sections. Ideally one wants to partition the graph along flowlines. This would totally physically minimize intercommunication between cells on different processors. However, if one knew the the flow lines ,the problem would be "solved", so that's not feasible. An approximation to this however might be a reasonable approach, but at the start one doesn't know even that.

The system in normal operation spreads the partitioned graph of triangulated cells across processors and their particles. Then a time step is run, particles are moved (collisionless) and moved across boundaries if necessary. Then they are collided. Every once in awhile we have to save state, this is done with the particles on each processor to a separate file so you don't serialize by writing to a single file on a network file system. These files may be collected later.

MPI-SEND-RECV is utilized throughout the simulation for communication. The undirected property of the graph is exploited in this way. They communications are ordered based on the min of the processor pair, and then ordered based on the second number. One send is done for all the particles that need to be sent to that processor. This imposes unnecessary ordering, but works. The number of particles to send is sent first and then a loop is done sending the particles.

Number of Processors	Speedup
1	1.0
2	1.7
4	2.7
8	3.2

Table 1: Parallelization Measurements.

3 Results

A simulated a flow over a serrated saw-tooth edge was performed. The example was provided by Ed Piekos. There were 200,000 particles in the system. The table describes the results obtained. Given the independence of the problem, it would seem that working more on the communication and load balancing could have some positive effect on scaling up the effective parallelization of this problem.

4 Conclusions and Future Work

I have implemented a first order attempt at parallelizing this simulation. In the future I plan to add the flow line based dynamic load balancing and also try to improve the sending strategy to not add artificial constraints.

References

[1] Ed S Piekos Micromechanical Device Sirect Simulation Monte Carlo Master's thesis, MIT, June 1995.