Assignment 0

Daniel Driver

February 7, 2013

1 About Me

I am currently a second year graduate student in Mechanical Engineering with a B.A. in physics from UC Berkeley. I am advised by Professor Tarek Zohdi in the Computational Materials Research Laboratory(CMRL). I am currently developing models and computational methods to simulate the manufacturing of flexible electronics with an emphasis on flexible photovoltaics(solar panels). With these tools, the hope is that manufacturing techniques can be developed to increase efficiency and reduce the cost of the electronics. More generally, I am interested in almost any engineering application including but not limited to topics in optics, solid-fluid interactions, automated design and topology optimization.

From CS267 I hope to gain considerable experience writing high quality parallel codes so that I can write my own software in the future and be confident that I am doing it reasonably well. While I enjoy learning about the low level aspects of parallelization like hardware and algorithms, I am most interested in how to use parallel computing for engineering applications.

2 Molecular Dynamics-GROMACS

GROMACS is an open source project designed to do molecular dynamics simulations with millions of particles[1]. The most prominent uses of the software package is for calculations of simple fluids and the biomechanics problems such as protein folding. GROMACS is a general purpose package so it is surely run on many top super computers. According to the NERSC website GROMACS is available on Hopper.

2.1 Load Balance

Molecular dynamics simulations are essentially the calculation of Newtonian mechanics in many body systems. Particles are engendered with forces laws which determines the force it applies on other particles. The forces from the surrounding particles are summed and then F=ma is integrated to find positions and velocities for the particles at later times. The most computationally intensive part of the MD simulation is the calculation of the forces between the particles. Parallelization is used to speed this up by binning the particles in the system and assigning the calculation to different processors. Because this calculation is the most intensive, GROMACS balances the load by making domains with volumes of particle that produce equal calculation times in this step. Imbalance is measured by how long a domain takes to compute compared to the other domains. A domain is modified to balance load by "scaling the relative lengths of the cells in each dimension with

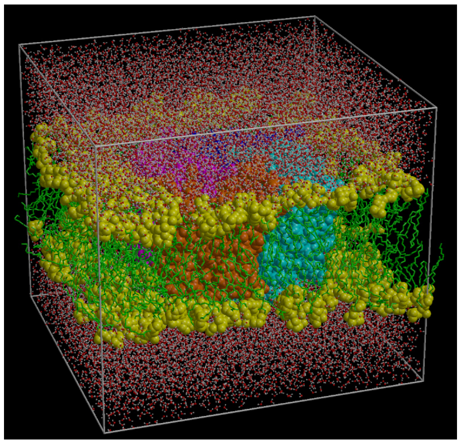

Figure 1: A visualization of an MD simulation found at http://www3.mpibpc.mpg.de/groups/de_groot/gallery.html

.5 times the load imbalance, and a maximum of 5%"[1] change. The programmers of GROMACS have found by experiment that this method of evolving domains leads to efficient and stable load balancing.

2.2 Memory Access

Memory access has the potential to be a bottle neck in MD simulations. Memory must be accessed at each step to calculate the relative forces between nearby particles. Since memory for spatially local particles is accessed at the same time it is efficient to store the data locally in memory also. However, even if the at the start of the simulation particles are store locally in memory relative to their position, particles move significantly during the course of a simulation and thus move to different domains. The result is particles not accessed at the same time are no longer stored locally in memory. Significant improvements in performance, especially when scaling up to many processors, is achieved by sorting, and storing particle data in memory by their spatial locality in the simulation when they become too disordered. In table 1 of the paper by Hess et al.[1], we see that this leads to a speed up of a factor of 2 in systems of size 2×10^5.

2.3 Network Communication

For the electrostatics calculation, a method called Particle Mesh Ewald(PME) is used. According to the paper[1] the method requires two fast Fourier Transform, an operation which requires global knowledge of the system. As such, each node must communicate with all other nodes at each step causing p^2communications were p is the number of nodes in the supercomputer

GROMACS discovered that the communication protocols standard to MPI are not conducive to the network architecture found in cluster supercomputers and is particularly pronounced on Beowulf clusters over high latency Ethernet networks. During the step to communicate with all other nodes, messages would be sent at the same time and overload the network interface cards of the nodes leading to dropped messages and thus the need to resend[2]. In systems with more than about 10 nodes this becomes the dominant bottleneck in the computation. GROMACS discovered that by managing the messages manually and sending messages in an order fashion that they could avoid network congestion.

References

[1] Berk Hess, Carsten Kutzner, David van der Spoel, and Erik Lindahl. GROMACS 4:Â Algorithms for Highly Efficient, Load-Balanced, and Scalable Molecular Simulation. *Journal of Chemical Theory and Computation*, 4:435–447, 2008.

[2] Carsten Kutzner, David Van Der Spoel, Martin Fechner, Erik Lindahl, Udo W. Schmitt, Bert L. De Groot, and Helmut GrubmÃŒller. Speeding up parallel gromacs on high-latency networks. *Journal of Computational Chemistry*, 28(12):2075–2084, 2007.