Parallel N-Body Problem

Girija Narlikar and Dafna Talmor *

girija@cs.cmu.edu, tdafna@cs.cmu.edu

School of Computer Science

Carnegie Mellon University

January 18, 1994

1 Introduction

We are interested in the parallel implementation of the Barnes-Hut N-Body algorithm. The Barnes-Hut method uses a simple hierarchical tree algorithm, and has numerous applications in areas such as astrophysics and fluid dynamics. We plan to use Nesl as a tool for implementing different methods of vectorizing the tree construction and traversal in this algorithm and measuring its efficiency.

2 The N-Body Problem

The classical gravitational N-Body problem simulates the evolution of a system of bodies under gravitational forces. Every body is modeled as a point mass and exerts forces on all other bodies in the system. The simulation proceeds over timesteps, each time computing the net force on every other body and thereby updating its position and other attributes. If all pairwise forces are computed directly, this requires $O(N^2)$ operations on each timestep. Hierarchical tree-based methods have been developed to reduce the complexity. One of the first few algorithms is the Barnes-Hut algorithm [3], which is $O(N \log N)$ for uniform distributions. There also exist more complex algorithms that are $O(N)$[4] for uniform distributions. A recent algorithm by Callahan [7] has a time complexity of $O(N)$ for general distributions.

*This research will be conducted with the participation of Guy E. Blelloch and Gary L. Miller

The Barnes-Hut algorithm is based on a hierarchical octree representation of space in three dimensions. The algorithm has two phases. The first phase consists of constructing the octree by adding particles to an initially empty root cell, and subdividing a cell into eight children when it contains more that one particle. In the second phase, the tree is traversed once per particle to compute the net force acting on it. We start at the root, and at each step, if the size of the cell is small compared to the distance of its center of mass from the particle, we use multipole approximation to compute the force on the particle due to the entire subtree under that cell. Otherwise, each of its subcells is visited.

The parallelism exploited in all phases is across particles, except in computing centres of mass of the cells, where it is across cells. A number of methods have been suggested, to vectorize the tree construction and traversal [5, 6].

3 Challenge Goals

There have been several efforts to implement N-Body code on parallel machines. The Stanford Splash benchmarks includes the Barnes-Hut algorithm as one of the applications[11]. They studied how to parallelize the code on a shared-memory model and derived speedups for up to 64 processors. Their algorithm has a serial bottleneck at the root of the tree, and is therefore not appropriate for a much larger number of processors. A similar version, for a distributed memory machine, was implemented by Salmon[9]. We plan to look at fully parallel implementations. Other researchers have implemented various other N-Body algorithms[2, 8, 10].

An important goal of our implementation will be to study the constants in the order of complexity of the algorithm, and to see how the complexity is effected in-practice by non-uniform distributions (the Barnes-Hut is $O(N \log N)$ for uniform distributions). We also hope to look at how the constants compare with other N-body algorithms, some of which are theoretically better (i.e. $O(N)$). Ideally we would like to interact with other researchers doing implementations of N-Body codes.

We aim to efficiently implement the tree-construction and tree-traversal operations in the Barnes-Hut N-Body algorithm. This algorithm has a similar tree-construction phase as in the All Nearest Neighbours algorithms, but differs in the fact that we do a tree-traversal instead of a tree search in order to calculate the force on a body due to the presence of the other bodies. Hence we hope to use common code wherever possible. However, instead of comparing various algorithms, we wish to concentrate on comparing different ways to vectorize the operations on trees.

Nesl [1] is a data parallel language that supports nested data parallelism. It presents to the programmer a PRAM model of computation. Therefore issues like load balancing and data distribution, which are critical to the efficiency of the algorithm, are left to the Nesl compiler to handle. This will enable us to concentrate on vectorization of the tree code, using Nesl as a tool for writing elegant, simple parallel algorithms.

We will use the same platforms and input sets for evaluation, as the ones we plan to use for the All Nearest Neighbour algorithms.

4 The Nearest Neighbours and N-body problems

The nearest neighbours problem and the N-body problems are closely related, so much that when a new algorithm is presented for one, a corresponding algorithm using the same technique closely follows for the other. Mathematically, the similarity between them can be brought forth by stating the nearest neighbours problem as computing for each point the function:

$$f(P_i) = \sum_{P_k \in P} \min Distance(P_i, P_k)$$

Where as the N-body problem can be stated:

$$f(P_i) = GM_{P_i} \sum_{P_k \in P} \frac{M_{P_k}}{Distance(P_i, P_k)^2}$$

This formulation corresponds to the naive $O(N^2)$ algorithms for the problems, which are reduced by similar geometric based divide and conquer techniques to $O(N \log N)$.

Therefore, when solving one problem, in essence we have a library of techniques and codes which can be used in the implementation of the other. Nevertheless, our research focus is different for each problem. In the Nearest Neighbours problem, we are interested in comparing the relative efficiency of the different theoretical algorithms in existence, and possibly of some practical hybrids. For the N-body problem, we are interested in an in-depth investigation into the implementation of one simple variant.

References

[1] Guy E. Blelloch. Nesl: A nested data-parallel language. Technical Report CMU-CS-93-129, CMU, School of Computer Science, April 1993.

[2] F.Zhao and S.L.Johnsson. The parallel multipole method on the connection machine. *SIAM Journal on Scientific and Statistical Computing*, 12(6):1420–1437, november 1991.

[3] J.E.Barnes and P.Hut. A hierarchical $O(N \log N)$ force calculation algorithm. *Nature*, 324(4):446–449, December 1986.

[4] L.Greengard and V.Rokhlin. A fast algorithm for particle simulation. *Journal of Computational Physics*, 73(325), 1987.

[5] L.Hernquist. Vectorization of tree traversals. *Journal of Computational Physics*, 87:137–147, 1990.

[6] J. Makino. Vectorization of a treecode. *Journal of Computational Physics*, 87:148–159, 1990.

[7] P.B.Callahan. Optimal parallel all-nearest-neighbours using the well-separated pairs decomposition. In *34th annual symposium on foundations of computer science*, pages 332–340, Palo Alto, 1993. IEEE.

[8] P.H.Mills, L.S.Nyland, J.F.Prins, and J.H.Reif. Prototyping N-body simulation in proteus. In *Proceedings Sixth International Parallel Processing Symposium*, pages 476–482, 1992.

[9] J. Salmon. Parallel $N \log N$ N-body algorithms and applications to astrophysics. In *COMPCON Spring '91, Digest of Papers*, pages 73–78, 1991.

[10] S.Bhatt, M.Chen, C.-Y.Lin, and P.Liu. Abstractions for parallel N-body simulations. In *Proceedings Scalable High Performance Computing Conference*, pages 26–29, 1992.

[11] J.Pal Singh, W.-D.Weber, and A.GUPTA. Splash: Stanford parallel applications for shared-memory. *Computer Architecture News*, 20(1):5–44, March 1987.