Parallel $N \log N$ N-body Algorithms and Applications to Astrophysics

John Salmon*
California Institute of Technology
Mail Code 206-49, Pasadena, CA 91125, USA

December 8, 1990

Abstract: A parallel version of the Barnes-Hut N-body algorithm is described. The algorithm first assembles a tree data structure which represents the distribution of bodies at all length-scales. A domain decomposition is used to assign regions of space, and hence bodies, to processors. An adaptive load balancing technique is used to insure that processors are assigned equal amounts of work. A tree is built in each processor, and after $\log_2 N_{proc}$ exchanges of data, each processor has a restricted version of the tree which is sufficient for force calculations on bodies which lie within its spatial domain. We have obtained a speedup of over 380 on a 512 processor Ncube system. Overhead is primarily due to redundant calculation and processor "waiting", i.e. time spent idle waiting for another processor to provide necessary data.

1 The BH algorithm.

The general N-body problem may be stated as the following set of 6N (in three dimensions) ordinary differential equations:

$$\frac{d\vec{x}_i}{dt} = \vec{v}_i \quad (1)$$

$$m_i \frac{d\vec{v}_i}{dt} = \sum_{j \neq i} \vec{F}_{ij} \quad (2)$$

In astrophysical simulations, the force term, F_ij is Newtonian gravity:

$$\vec{F}_{ij} = \frac{G m_i m_j \vec{r}_{ij}}{|r_{ij}|^3}. \quad (3)$$

The gravitational force is "long-range", meaning that there is no cutoff, beyond which the force may be considered negligible. In principle, it is necessary to evaluate the entire sum on the right-hand side of Eqn. 2 at each timestep of the time integration. Naively, this requires $O(N^2)$ operations on each timestep.

*This work was supported by the U.S. Department of Energy: Applied Mathematical Sciences (Grant DE-FG03-85ER25009); Program Manager of the Joint Tactical Fusion Program Office; U.S. National Science Foundation: Center for Research on Parallel Computation (Grant CCR-8809615).

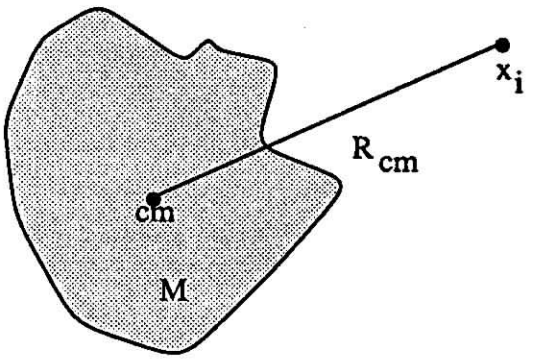

Figure 1: A collection of bodies may be approximated by a point mass located at the center of mass.

The Barnes-Hut (BH) [1] algorithm is one of a number of algorithms [1-5] which use a multipole expansion and a hierarchical data structure to reduce the complexity of computing long-range interactions like gravity. The multipole expansion allows one to treat a collection of bodies as a point mass (perhaps with quadrupole and higher moments) located at the center of mass. In Figure 1, the force on point x_i may be evaluated approximately as:

$$\vec{F}_i = \sum_j \frac{G m_i m_j \vec{r}_{ij}}{|r_{ij}|^3} \approx \frac{G m_i M \vec{R}_{cm}}{|R_{cm}|^3} \quad (4)$$

The quality of the approximation in Eqn. 4 is a decreasing function of the ratio: $b/|R_{cm}|$, where b is the radius of the collection of bodies.

In the BH algorithm, multipole moments are computed for cubical cells an octree of variable depth. The tree is constructed *ab initio* on each timestep with the following properties:

1. The root cell encloses all of the bodies.
2. No terminal cell contains more than m bodies.
3. Any cell with m or fewer bodies is a terminal cell.

A typical two-dimensional BH tree with $m = 1$ is shown in Figure 2.

To compute the force on a body, one traverses the tree starting at the root. Any time one encounters a

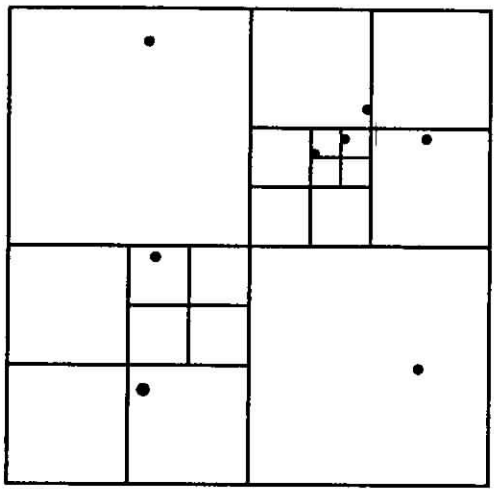

Figure 2: An $m = 1$ BH tree.

cell with a sufficiently small value of $b/|R_{cm}|$, one uses the multipole approximation. Thus, distant cells, which comprise many individual bodies, may be approximated in unit time. The resulting algorithm, when applied to all bodies, requires $O(N \log N)$ operations to evaluate the forces on all N bodies.

2 Parallelization.

Developing a parallel implementation of the BH algorithm proceeds in three steps:

1. Domain decomposition. Divide space into rectangular regions with one processor assigned to each region. All bodies that lie within that region are the "responsibility" of that processor.

2. Build a "locally essential" version of the BH tree in each processor. This is a subset of the full tree which is sufficient for all force calculations within the processor's restricted spatial domain.

3. Proceed with the force-evaluation algorithm exactly as in the serial case, i.e. traverse the locally essential tree once for each body.

2.1 Domain decomposition.

If a given multipole approximation is sufficient for a particular particle, then there is a very good chance that it is sufficient for nearby particles as well. This fact suggests that there is considerable advantage in processors being responsible only for bodies in a restricted spatial domain. If a processor is guaranteed to only compute forces within a restricted domain, then it only needs to build and record a limited subset of the entire BH tree. A good domain decomposition will divide the total work load equally amongst the processors, and minimize (or

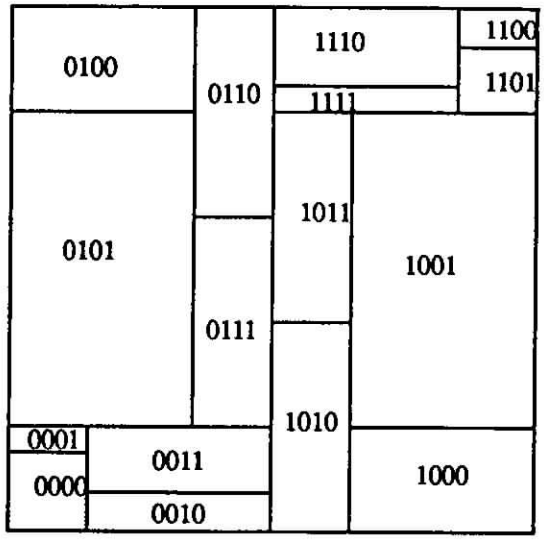

Figure 3: Orthogonal recursive bisection of space, with processor numbers assigned.

at least maintain an acceptably small level of) communication.

Assigning spatial domains to processors is accomplished by orthogonal recursive bisection [6]. Both space and the set of processors are divided into halves, and each half of the spatial domain is assigned to one half of the processors. The bisection of space is performed on alternating cartesian directions, until exactly one processor is assigned to one region of space. A typical decomposition resulting from orthogonal recursive bisection is shown in Figure 3. Note that the "tree-like" structure shown in Figure 3 is logically distinct from the BH tree.

In order to achieve a well balanced load, the domain decomposer must be able to estimate the partition of work that results from a partition of space. Astrophysical N-body simulations are usually highly irregular, consisting of one or more galaxies with very dense cores and diffuse haloes. The spatial distribution of bodies is highly non-uniform. In addition, because of the large range of densities, the depth of the BH tree is highly non-uniform also, which means individual particles contribute different amounts of work to the total load. Nevertheless, the numerical stability of the time-integrator requires that the time-step be sufficiently small that the local density of particles cannot change very much from one time-step to the next. Thus, the work required to update a particular particle cannot change significantly from one time-step to the next, although it may be very different from one particle to the next. This allows us to estimate the work associated with a given domain simply by estimating the time (measured either by counting interactions or by reading the on-board clock) required to update each of the particles in that domain, on the last time-step. The coordinate of each split in the or-

thogonal recursive bisection is determined by iteratively seeking a solution to

$$W(x_{split}) = 0 \quad (5)$$

$$W(x) = \frac{w_>(x) - w_<(x)}{w_>(x) + w_<(x)} \quad (6)$$

where $w_<(x)$ and $w_>(x)$ are the work associated with particles with coordinate less than x and greater than x, respectively, estimated from the last time-step. The iterative root finder need not converge to very high accuracy. Since the estimated work is only accurate within a few percent, it is a waste of time to seek a solution of Eqn. 5 which is accurate to better than a few percent.

Additional load imbalance arises because choice of splitting coordinates based on Eqn. 5 only balances the load that can readily be assigned on a per-particle basis. The computation associated with building the BH tree is not accounted for by Eqn. 5. One possibility is to develop an empirical model of the tree-building time as a function of the number density of bodies, etc, and to use this model in place of $w_<(x)$ and $w_>(x)$. A much better approach is to rely on the fact that whatever the exact relationship between the total work and the per-particle work, it must satisfy the following two properties:

1. It is monotonic in the per-particle work.

2. It does not change significantly from one time-step to the next.

Then we can seek a split point, x_{split} which is not exactly the median of the per-particle work distribution:

$$W(x_{split}) = p_{new} \quad (7)$$

$$p_{new} = p_{old} - I_{old} * \omega \quad (8)$$

$$I_{old} = \frac{\bar{w}_>(x) - \bar{w}_<(x)}{\bar{w}_>(x) + \bar{w}_<(x)} \quad (9)$$

$$0 < \omega < 1 \quad (10)$$

where $\bar{w}_>(x)$ and $\bar{w}_<(x)$ are the measured total work performed by processors assigned to the upper and lower regions on the last time-step. I_{old} is the actual load imbalance encountered on the last time-step. The parameter ω is arbitrary. We have found that $\omega = 0.75$ results in convergence to good load balance in two or three time-steps, and prevents oscillations.

2.2 The locally essential tree.

The next step is to communicate information from processor to processor so that each processor has sufficient data to update the bodies in its domain. This technique is a generalization of that used in more regular domain decompositions, in which a "guard layer" is exchanged between neighboring processors, e.g., in a finite difference calculation [7]. Here, the equivalent of the guard layer is a representation of the tree at successively coarser levels of detail as one moves away from the processor's boundary.

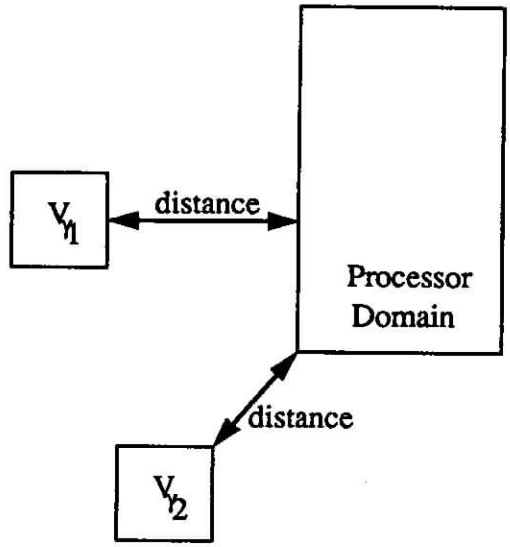

Figure 4: Distance used in domain opening criterion predicate.

The natural communication topology imposed by orthogonal recursive bisection is that of a hypercube, with each division of space corresponding to one of the dimensions of the hypercube. This hypercube topology can be exploited to broadcast data at exactly the correct level of detail to all of the processors.

The procedure is as follows:

```
Build a BH tree from local particles
for (each bisection from ORB)
    enqueue for transmission data which
        may be necessary for any domain on
        the other side of the bisection.
    exchange enqueued data with a processor
        on other side of the bisector.
    insert the received data into the
        local BH tree.
endfor
```

It is easy to show by induction that after this process is complete, each processor has exactly the data it will need to compute the forces on any body within its domain.

The algorithm described above requires that one be able to answer the "domain opening criterion" question– is a given cell's multipole approximation sufficiently good for any body in a given rectangular domain. This question is readily answered by comparing the size of the cell to the distance shown in Figure 4.

We have found that a slight modification of the original BH accuracy criterion has two beneficial effects [8]:

1. Eliminates a source of systematic error.

Machine	nodes	time(sec)	N
MarkIIIfp	64	280	1.1 million
Ncube1	512	300	200,000
Ncube2	512	80	1.1 million
Intel gamma(C)	64	290	1.1 million
" (asm, est.)	64	50-75	1.1 million

Table 1: Timing results for several parallel processors.

2. Reduces the number of overly conservative answers to the domain opening criterion predicate.

2.3 Evaluate the forces.

Since each processor has a subset of the BH tree which is sufficient for its own particles, there is no more need for interprocessor communication. In fact, the original serial code for force evaluation may be used completely unchanged. The parallel algorithm will produce results identical to the serial algorithm, except for a very small amount of roundoff error which results from the non-associativity of floating point operations.

3 Performance.

Our most extensive series of performance measurements were carried out on the 512 processor Ncube1 at Caltech. Each processor of this machine has only 512kbytes of memory, which barely allows us to explore the large grain-size limit, and restricts the total number of bodies we could simulate. The program is written with Cubix [9] so there is no separate host program to interfere with running on serial machines. Typically, analysis and debugging of algorithmic changes was done on a serial machine running UNIX. We have also run the program on several other distributed memory parallel machines. Some timings are shown in Table 1. The extensive series of timings on the NCube are discussed in the following sections.

3.1 Overhead.

It is common practice to report performances of parallel machines in terms of speedup, S, or efficiency, ϵ.

$$S = \frac{T_{oneproc}}{T_{N_{proc}}} \tag{11}$$

$$\epsilon = \frac{S}{N_{proc}} \tag{12}$$

The overhead, f, which is algebraicly related to the efficiency is somewhat more useful:

$$f = \frac{1}{\epsilon} - 1 = \frac{T_{N_{proc}} - T_{intrinsic}}{T_{intrinsic}} \tag{13}$$

$$T_{intrinsic} = \frac{T_{oneproc}}{N_{proc}} \tag{14}$$

The overhead may be readily separated into components which combine additively to give the total overhead, i.e.,

$$f = f_{cplx} + f_{wait} + f_{comm} + f_{imbal}. \tag{15}$$

The complexity overhead, f_{cplx}, arises from the additional work required by the parallel implementation over and above that required by a serial implementation. The waiting overhead arises from delays introduced at synchronization points, i.e., blocking communication routines and explicit synchronizations. Communication overhead, f_{comm}, arises from the time required to exchange data between processors. It is the time spent in actual communication, after the synchronization point has been passed. Finally, f_{imbal} measures the difference in speed between the slowest processor and the average.

3.2 Models.

The performance of the serial BH algorithm is known to depend on the detailed distribution of points in space. Thus, the parallel performance was measured with three types of data.

1. A uniform sphere, shown in the figure connected by dashed lines.

2. A Jaffe sphere, [10] which is highly concentrated near the origin, shown in the figure connected by solid lines.

3. A snapshot of a merger simulation [11] shown in the figures connected by dotted lines.

3.3 Data.

The speedup, S, and total overhead, f, are shown in Figures 5 and 6. Overheads are plotted against the grain-size,

$$N_{grain} = \frac{N}{N_{proc}}, \tag{16}$$

which demonstrates that the overhead decreases as more bodies are added. Evidently, the speedup flattens out as a function of N_{proc} at some point for a fixed number of bodies. Nevertheless, for large grain sizes, the overheads are quite small.

The four components of overhead are shown in Figures 7 through 10. All the overheads decrease with increasing grain size, which means we can expect improved performance from newer, larger machines. This prediction is confirmed by the timings in Table 1. Interestingly, communication overhead is the least significant component of the overhead, in contrast to the majority of parallel scientific algorithms. The largest component of the overhead is the complexity, which arises from redundant computation of multipole moments during the construction of the tree.

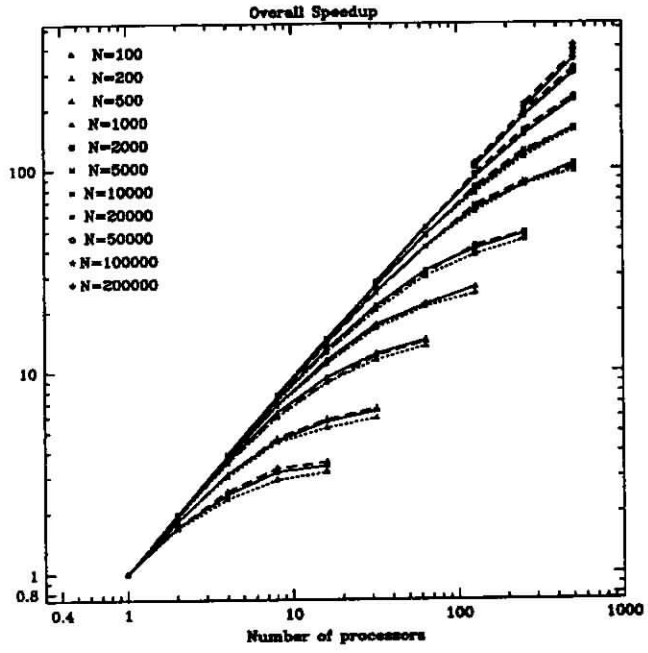

Figure 5: Speedup vs. number of processors.

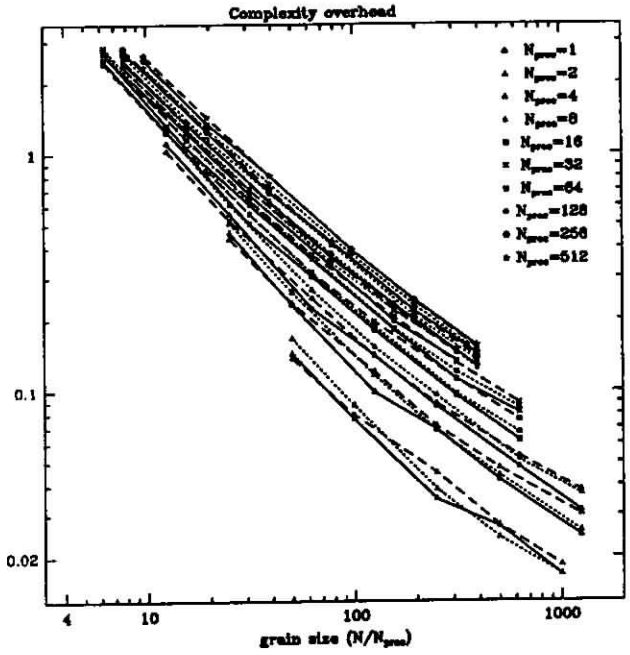

Figure 7: Complexity overhead vs. grain size.

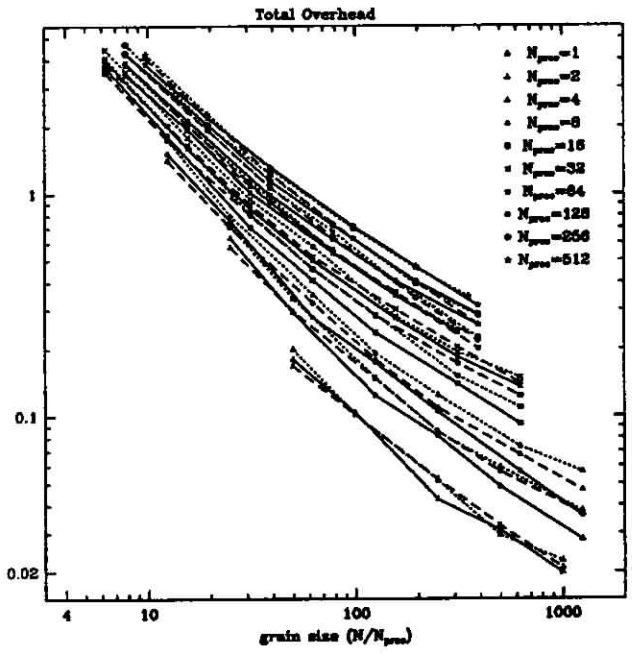

Figure 6: Overhead vs. grain size.

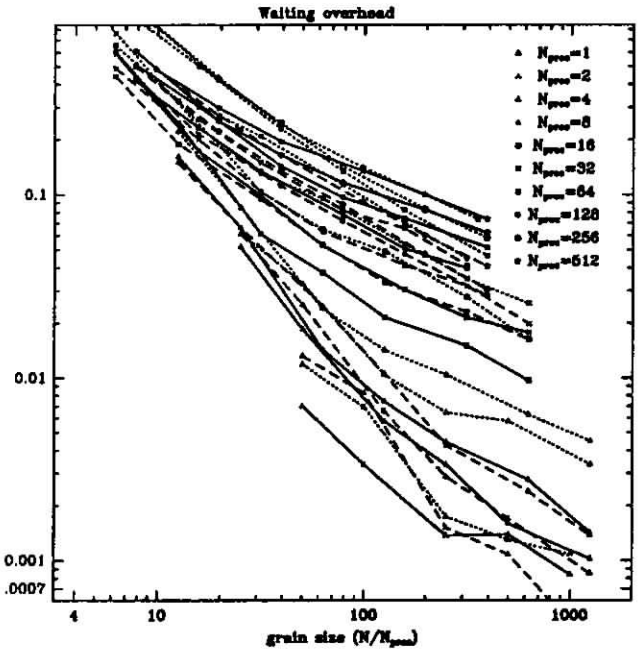

Figure 8: Waiting overhead vs. grain size.

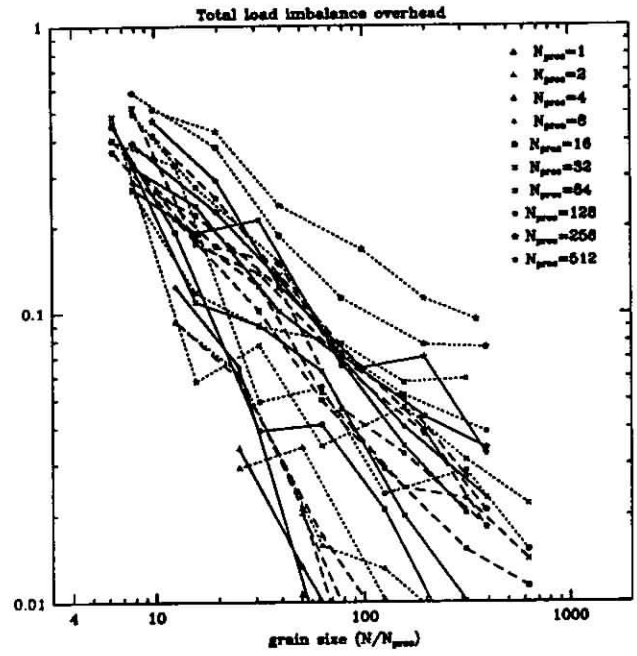

Figure 9: Imbalance overhead vs. grain size.

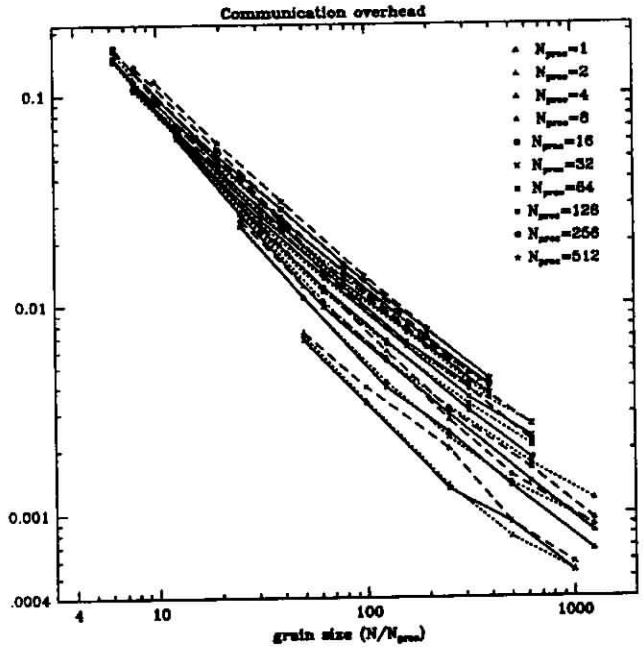

Figure 10: Communication overhead vs. grain size.

References

[1] J. Barnes and P. Hut, "A Hierarchical O(NlogN) Force-Calculation Algorithm," *Nature*, **324**, 446-449, 1986.

[2] A.W. Appel, "An efficient Program for Many-Body Simulation," *SIAM J. Sci Stat. Comput.*, **6**, 85, 1985.

[3] L. Greengard, "The Rapid Evaluation of Potential Fields in Particle Systems," PhD Thesis, Yale University, Computer Science Research Report YALEU/DCS/RR-533, 1987.

[4] J.G. Jernighan and D.H. Porter, "A Tree Code with Logarithmic Reduction of Force Terms, Hierarchical Regularization of All Variables and Explicit Accuracy Controls," *Astrophysical Journal (Suppl.)*, 871, 1989.

[5] W. Benz, R.L. Bowers, A.G.W. Cameron, and W.H. Press, "Dynamic Mass Exchange in Doubly Degenerate Binaries I. 0.9 and 1.2 Msolar Stars," *Astrophysical Journal*, **348**, 647, 1990.

[6] G.C. Fox, "A Graphical Approach to Load Balancing and Sparse Matrix Vector Multiplication on the Hypercube," *Numerical Algorithms for Modern Parallel Computer Architectures*, ed. M. Schultz, Springer-Verlag, pp. 37-62, 1988.

[7] G.C. Fox, M. Johnson, G. Lyzenga, S. Otto, J. Salmon, and D. Walker, *Solving Problems on Concurrent Processors*, Prentice Hall, Englewood Cliffs, NJ, 1988.

[8] J. Salmon, "Parallel Hierarchical N-body Methods," PhD Thesis, California Institute of Technology, 1991.

[9] J. Salmon, "CUBIX: Programming Hypercubes Without Programming Hosts," *Hypercube Multiprocessors 1987*, ed. M.T. Heath, SIAM, Philadelphia, pp. 3-9, 1987.

[10] W. Jaffe, "A Simple Model for the Distribution of Light in Spherical Galaxies," *Mon. Not. Roy. Astron. Soc.*, **202**, 995, 1983.

[11] J. Salmon, P. Quinn, and M. Warren, "Using Parallel Computers for Very Large N-body Simulations: Shell Formation Using 180K Particles," *Proceedings of 1989 Heidelberg Conference on Dynamics and Interactions of Galaxies*, ed. R. Wielen, Springer-Verlag, 1989.