Astrophysical N-body Simulations Using Hierarchical Tree Data Structures

1992 Gordon Bell Prize Winner.

Michael S. Warren

Theoretical Astrophysics
Los Alamos National Laboratory
Los Alamos, NM 87545

John K. Salmon

Physics Department
California Institute of Technology
Pasadena, CA 91125

Abstract

We report on recent large astrophysical N-body simulations executed on the Intel Touchstone Delta system. We review the astrophysical motivation, and the numerical techniques, and discuss steps taken to parallelize these simulations. The methods scale as $O(N \log N)$, for large values of N, and also scale linearly with the number of processors. The performance, sustained for a duration of 67 hours was between 5.1 and 5.4 Gflop/sec on a 512 processor system.

1 Astrophysical and numerical background.

The process by which galaxies form is undoubtedly among the most important unsolved problems in physics. There is a wealth of observational data, whose quality and quantity is ever increasing. Modern observations span the electromagnetic spectrum from radio frequency to gamma rays. Unfortunately, we still lack a firm theoretical understanding of the images put on photographic plates 40 years ago. The answer to a question as simple as, "Why are there two families of galaxies, spiral and elliptical?" still remains a mystery.

Astrophysics is at a disadvantage to some of the more terrestrial sciences. It is simply impossible to conduct experiments on galaxies. An investigator can easily change the recipe for making a superconductor. On the other hand, the recipe for making a galaxy requires 10^{45} grams of matter, and several billion years of "baking", which is far beyond the patience of most scientists. With the use numerical methods, however, one can *simulate* the the behavior of 10^{47} grams of matter over a span of 10^{10} years.

The physical laws governing the evolution of gravitationally interacting particles are quite simple. Given initial positions and velocities, all that is required are Newton's laws of motion and universal gravitation. The principles are clear, but the intrinsic non-linearity of the equations has limited analytic studies to small perturbations or restricted symmetries. The only known way to obtain accurate three-dimensional solutions is via numerical integration.

The first step in understanding how galaxies and stars form is to understand the environment in which their formation occurs. Thus, we use simulations to study the shapes and dynamics of "dark matter" halos which are known to surround observed galaxies. For our purposes, the precise nature of the dark matter is unimportant; the fact that it interacts only through dissipationless gravitational forces allows us to accurately simulate its evolution. Since dark matter constitutes the bulk of the mass of the Universe, it plays a dominant role in the dynamical processes involved in galaxy formation.

The dark matter halos that we simulate are not directly observable, but they provide the gravitational potential well into which normal matter falls, forms stars, and emits photons which are detected by astronomers.

The fundamental physical equation governing the dynamics of stars, dark matter and other "dissipationless" phenomena is the Boltzmann equation:

$$\partial_t f(\vec{x},\vec{v},t) + \vec{v}\cdot\partial_{\vec{x}} f(\vec{x},\vec{v},t) + \vec{a}\cdot\partial_{\vec{v}} f(\vec{x},\vec{v},t) = 0, \qquad (1)$$

which describes the evolution of a phase-space density function, f, with seven independent variables, $\vec{x}$, $\vec{v}$, t. It is common to treat such systems on computers by *particle methods* or *N-body methods*[1], rather than finite-element or finite-difference methods, which cannot cope with the high dimensionality of the problem domain. In an astrophysical N-body simulation, the phase-space density distribution is represented by a large collection of "bodies" (labeled by the indices i, j) which evolve in time according to the laws

of Newtonian physics:

$$\frac{d^2\vec{x}_i}{dt^2} = \sum_{j\neq i}^{N} \vec{a}_{ij} = \sum_{j\neq i} -\frac{Gm_j\vec{d}_{ij}}{|d_{ij}|^3}, \qquad \vec{d}_{ij} \equiv \vec{x}_i - \vec{x}_j. \tag{2}$$

N-body simulations are essentially statistical in nature. More bodies means a more accurate and complete sampling of the phase space, and hence more accurate or more complete results. Astrophysical simulations require very large numbers of bodies because of the very large density contrasts that must be treated. Interesting phenomena occur in the dense cores of galaxies, as well as on the fringes and in the voids between galaxies. For example, if one is to study the shapes of isophotes or the formation of "shells" in the outskirts of halos, one must have enough particles in each halo to measure the isophotes with sufficiently low statistical noise.

Two factors govern the final state of a dynamical system like Eqn. 2: the equations of motion that describe the evolution, and the initial conditions which are provided as input. In the case of astrophysical N-body simulations, an appropriate set of initial conditions is a set of initial positions, velocities and masses for all of the bodies in the system. It is through these initial conditions that most of the physical hypotheses enter the system.

At the present epoch, the Universe is obviously highly irregular. Galaxies and stars as well as scientists and their supercomputers constitute huge, non-linear deviations from the large-scale uniformity in the distribution of matter. Numerous observations (perhaps the most compelling is the remarkable uniformity of 3° K cosmic background radiation) imply that the matter in the universe was far more uniformly distributed in the distant past. This is fortunate because it allows us to use linear approximations and analytic methods to study the evolution of matter in the early universe. Different hypotheses about the nature of the dark matter (e.g., cold or hot), the origin of the fluctuations (e.g., quantum fluctuations in an inflationary early universe), and the values of various global parameters, (e.g., the fraction of the mass of the Universe made up of baryons) can be folded together to produce statistical descriptions of the fluctuations that prevailed until the growth of fluctuations became non-linear.

These descriptions generally take the form of a power spectrum. By using Fourier transform techniques, we can create instances of particular power spectra in the form of particle positions, velocities and masses, i.e., in a form suitable as initial conditions for Eqn. 2. Then, by using numerical techniques, we can integrate Eqn. 2 until the present day, and compare the results *statistically* with observation (subject to the caveat that one cannot directly observe the dark matter which is simulated).

Direct implementation of the system of equations in Eqn. 2 is a trivial programming exercise. It is simply a double loop. It vectorizes well and it parallelizes easily and efficiently. It has been used for many years to study astrophysical systems. Unfortunately, it has an asymptotic time complexity of $O(N^2)$. Each of N left-hand-sides is a sum of $N-1$ right-hand-sides. The fact that the execution time scales as N^2 precludes the use of Eqn. 2 for values of N larger than a few tens of thousands, even on the fastest parallel supercomputers.

A number of *approximate* methods have been used which reduce the overall time, and allow simulation of systems with larger values of N. Methods employing an adaptive tree data structure have been popular in recent years because the resulting time complexity is $O(N \log N)$ or $O(N)$, and is relatively insensitive to the spatial distribution of particles [2, 3, 4]. N-body simulations which use adaptive tree data structures are referred to as *treecodes*.

The fundamental approximation employed by treecodes may be stated as:

$$\sum_j \frac{Gm_j\vec{d}_{ij}}{|d_{ij}|^3} \approx \frac{GM\vec{d}_{i,cm}}{d_{i,cm}^3} + \ldots, \tag{3}$$

where $\vec{d}_{i,cm} = \vec{x}_i - \vec{x}_{cm}$ is the vector from $\vec{x}_i$ to the center-of-mass of the particles that appear under the summation on the left-hand side, and the ellipsis indicates quadrupole, octopole, and further terms in the multipole expansion. Generally, we use the quadrupole approximation, which requires a second term on the right-hand side. The monopole approximation, i.e., Eqn. 3 with only the first term on the right-hand side, was known to Newton, who realized that the gravitational effect of an extended body like the Earth (consisting of some 3×10^{51} protons and neutrons) can be approximated by replacing the entire system by a point-mass located at the center of mass. The resulting simplification is enormous: a factor of 10^{51} if one wishes to know the force on a falling apple or satellite.

The approximation of Eqn. 3 is not always valid. In fact, it doesn't converge at all if the point at which the force is desired is inside a sphere that circumscribes the other masses. Generally, the approximation is better if the measurement point is far from the other masses. The scale by which one should judge "near" and "far" is simply the diameter of the sphere that encloses the mass points.

At this point it is convenient to introduce a data structure that represents the distribution of matter on all length-scales. Suppose the particles are organized into a spatial, *oct-tree*. The root of the tree corresponds to a cube that encloses all the bodies in the simulation. Non-terminal nodes in the tree have up to eight daughters, corresponding to the eight sub-cubes that result from cutting the parent in half in each dimension. Any cell containing only one body is

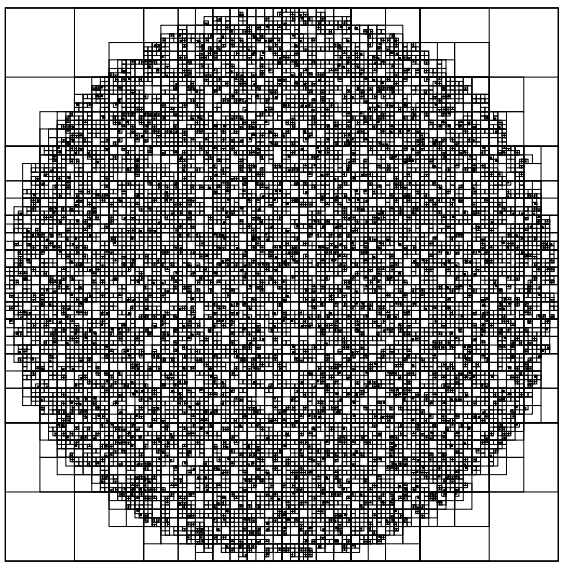

Figure 1: A two-dimensional tree with 10000 bodies uniformly distributed on the unit-disk.

terminal. A two-dimensional example is shown in Fig. 1. Furthermore, suppose that each internal cell in the oct-tree contains the total mass of the bodies in it, their center-of-mass and their quadrupole moments. Then it is possible to compute the force on any body by recursively traversing the tree according to the procedure outlined in Fig 2. Although it is not obvious from this brief description, the number of times Eqn. 3 is evaluated during a traversal of the tree is proportional to $\log N$. Computing the forces on all N bodies requires traversing the tree N times and is thus $O(N \log N)$ in time.

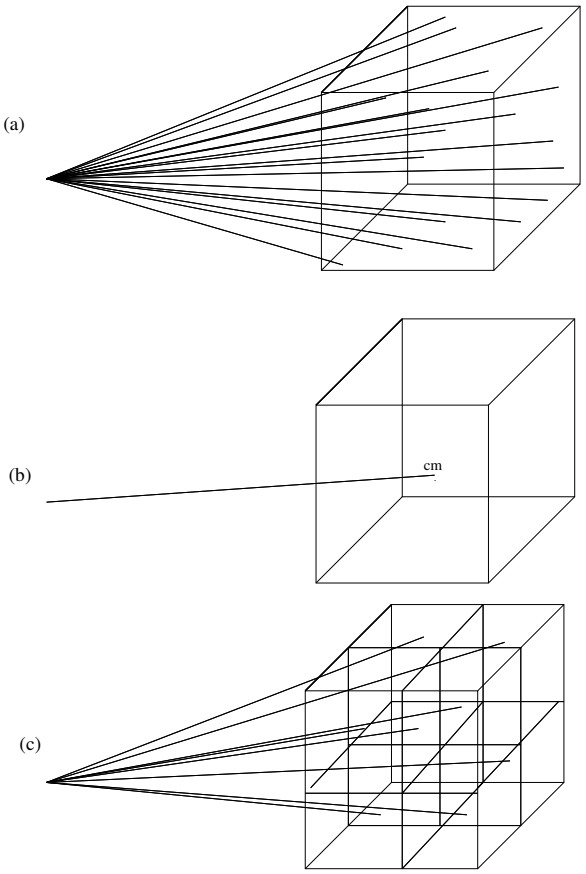

Figure 2: Schematic representation of the recursive structure of a treecode. (a) The exact force on a body is the result of a summation over all particles in a cell. (b) If the multipole approximation is valid, then the summation can be replaced by a single evaluation of Eqn. 3. (c) Otherwise, the cell is subdivided into eight daughters, and the procedure, i.e., steps (b) and (c), is recursively applied to the daughters.

2 Parallelism.

Astrophysical treecodes represent a formidable challenge for parallel computation. The difficulties stem from some fundamental properties of the problem:

- The distribution of bodies is highly non-uniform.
- The distribution of bodies is dynamic.
- The data structures are adaptive, and moderately complicated.
- Each body needs both global and local data for its update.

The non-uniformity of the data precludes use of a regular spatial decomposition. Instead, we adopted the technique of *orthogonal recursive bisection*, ORB [5], whereby space is recursively divided in two, and half the processors are assigned to each domain until there is one processor associated with each rectangular domain. In order to avoid wasting time due to load imbalance and idle processors, it is crucial to divide space into domains with equal workloads. The fact that the simulations are dynamic makes it impossible to precompute the decomposition or the communication pattern as would be the case with an irregular but static problem.

Complex data structures present problems as well. Standard notations like Fortran90 or High Performance Fortran cannot readily represent a distributed adaptive tree. More importantly, exactly what one means by a "distributed adaptive tree" is not immediately obvious. It certainly depends on what one intends to do with the data structure. An an-

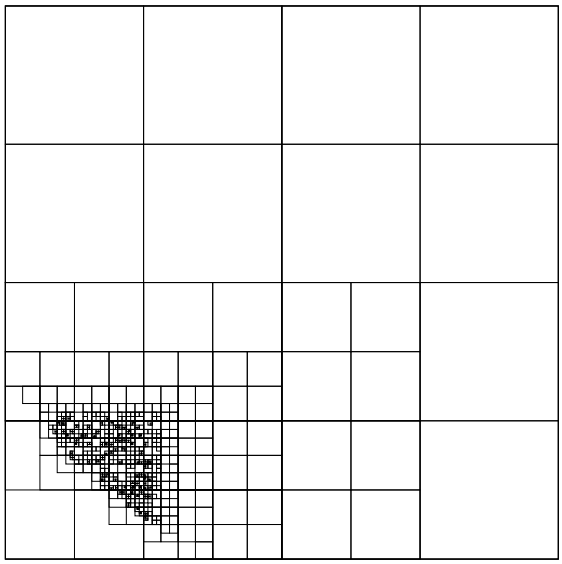

Figure 3: The locally essential tree for a processor whose domain is in the lower left corner of the system which gives rise to Fig. 1.

swer comes from consideration of the last difficulty: the need for both global and local data in the update of each body.

Specifically, every body *sees* only a fraction of the complete tree. The distant parts are seen only at a coarse level of detail, while the nearby sections are seen all the way down to the leaves. The crucial observation is that nearby bodies *see* similar trees. In fact, if one considers all the bodies in an ORB domain, one can construct the union of all the trees they *see*. This union is called the *locally essential tree*. It is the data that will be required to compute the forces on every body in the domain. A locally essential tree, for a processor whose domain is the lower-left corner of Fig. 1 is shown in Fig. 3. A strategy is now clear, at least in principle:

- Obtain the locally essential tree in every processor.
- Proceed exactly as in the sequential case, traversing the tree once for each particle.

The fact that the second step is exactly the same as in the sequential case is significant. It constitutes the bulk of the time used by the algorithm, and it is useful to be able to write highly tuned sequential assembly language without regard to communication, synchronization or other parallel issues.

The problem now is to obtain the locally essential tree. The key is to use the recursive structure provided by the ORB. The result of ORB is a set of $\log_2 P$ spatial bisectors and corresponding partitions of processors. Locally essential data can be routed to its correct location by looping over the bisectors, and repeatedly determining which local data is part of a locally essential tree on the *other side* of the cut. Every processor identifies its own exportable data, and then exchanges that data with a processor in the complimentary partition on the other side of the bisector. After $\log_2 P$ exchanges, every processor is in possession of its locally essential tree.

The determination of which data to send is intimately related to how the data will be used. It is crucial that one can determine, *a priori*, which parts of a tree in local memory will also be needed in the evaluation of forces on an unspecified remote processor. Returning to the original algorithm, we find that this determination is, in fact, possible. The sequential algorithm asks: Is this multipole acceptable for the evaluation of the force at a particular location? The parallel algorithm must also ask: Is this multipole acceptable for the evaluation of the force at any point in a rectangular parallelepiped (i.e., in a processor's domain)? The sequential algorithm prescribes that the first question is answered by computing a ratio of distance to cell-size. We can answer the second by using the same criterion with a redefined distance: the distance from the edge of the cell to the boundary of the rectangular domain.

3 Recent simulations.

Our parallel N-body code has been evolving for several years, and on several platforms. At press time, the fastest available platform is the Intel Touchstone Delta at Caltech, on which the simulations reported here were run. We have also run significant simulations on an Intel ipsc/860, Ncube machines, and the Caltech/JPL Mark III [6, 7, 8, 9, 10, 11].

The statistics quoted below are based on internal diagnostics compiled by our program. Essentially, we keep track of the number of interactions computed. Each monopole interaction costs 29 flops, each quadrupole interaction costs 77 flops and each evaluation of the opening criterion costs 6 flops (using the Livermore Loops prescription of 1 sqrt = 4 flops, 1 division = 4 flops and 1 comparison = 1 flop). The flop rates follow from the interaction counts and the elapsed wall-clock time. The flop counts are identical to the best available sequential algorithm. We *do not* count flops associated with decomposition or other parallel constructs. The reported times are for the entire application, including I/O, communication, program initialization, etc. Overall, we see about 15% of the time spent in parallel overhead (communication, load imbalance, etc.), which corresponds to a parallel efficiency of about 87%.

3.1 An 8.78 million body simulation.

In March 1992, we ran a simulation with 8,783,848 bodies on 512 processors for 780 timesteps. The simulation was of a spherical region of space 10 Mpc (Megaparsec) on a side; a region large enough to contain several hundred typical galaxies. Our simulation ran continuously for 16.7 hours, and carried out 3.24×10^{14} floating point operations, for a sustained rate of 5.4 Gflop/sec. Had we attempted the same calculation with a conventional $O(N^2)$ algorithm, it would have taken almost 3000 times as long to obtain an equivalent answer. We created 15 checkpoint files totaling 4.21 Gbytes. Had we saved all the potentially useful intermediate data, it would have required 273 Gbytes. A single checkpoint from this simulation exceeds 280 Mbytes. It is impractical to analyze the results on anything other than a parallel supercomputer. To that end, we have ported some of our analysis software to the Delta, and have isolated individual halos for further analysis. The 700 individual halos in this simulation have anywhere from 1,000 to 130,000 bodies, making them comparable in size to "state-of-the-art" isolated halo models running on Crays and other vector supercomputers. The Delta allowed us to evolve several hundred large halos simultaneously, in a realistic environment, providing us with much needed statistics, as well as information concerning environmental effects on evolution which cannot be obtained from isolated halo models.

3.2 Two 17.15 million body simulations.

In June 1992, in response to the recently announced measurement of the microwave background anisotropy by the COBE satellite [12, 13], we ran two large simulations of the Cold Dark Matter model of the Universe. The COBE measurement has constrained the last remaining free parameters left in this popular theory, and allows us to produce initial conditions in which the power spectrum of initial fluctuations is completely specified, including the overall amplitude. These are, by any measure, the largest N-body simulations ever run [1]. The results of the simulations are being analyzed. When compared with other observational constraints we hope they will provide compelling evidence either for or against the model

The simulations represented regions with diameter 250 Mpc and 100 Mpc, and had 17,158,608 and 17,154,598 bodies, respectively. The individual particles each represented about $3.3 \times 10^{10}\,M_\odot$ and $2.0 \times 10^{9}\,M_\odot$, respectively, so that galaxy-size halos are expected to form with tens to thousands of individual particles; enough to obtain reasonable statistics concerning the distributions and correlations of sizes. The spatial resolution was 20 kpc in both cases. They ran for 597 and 667 timesteps, in 23.5 and 28.6 hours, respectively, and wrote 21 and 27 data files for a total of 11.53 and 13.72 Gbytes. They respectively performed 4.33×10^{14} and 5.32×10^{14} floating point operations, sustaining rates of 5.2Gflop/sec and 5.1Gflop/sec.

[1] A simulation with 17 million bodies has been reported [14] using a different approximation which suffers from limited spatial resolution. It ran for over 600 hours on an IBM vector supercomputer. In contrast, our simulation, with marginally larger N, had twenty times the linear spatial resolution, and ran in one twentieth the time.

4 Supporting software.

High performance computing is not easy. The i860 processor promises extraordinary performance, but compiled code rarely if ever achieves that performance. Unfortunately, the field of N-body simulations does not possess a recognized set of abstractions like the BLAS for linear algebra. Thus, there is no highly tuned commercial software available for carrying out the inner loops of our calculations. In the quest for performance, we have written the inner force calculation loops entirely in i860 assembly language. Two routines were necessary, for monopole and quadrupole interactions, which constitute roughly 1000 lines of machine instructions. By arranging the calculations to proceed three at a time the code was fully pipelined, even including a Newton-Raphson inverse square root. In this manner, we achieve at least one addition or multiplication per clock cycle (notwithstanding cache delays). The assembly coded monopole and quadrupole interaction routines run at about 22 Mflops/sec/node; a factor of eight better than obtained with the C compiler.

Portability is an extremely important issue. The Delta is neither the first nor the last parallel supercomputer that will run our N-body code. We have found that we can achieve an acceptable degree of portability by disciplined use of a few very simple communication primitives (essentially the *cshift* routine from CrosIII [15]) and a well-defined model of I/O (essentially *cubix* [16] with some modifications). We have developed our own ANSI C implementation of these primitives, and we have ported them to all of the parallel processors at our disposal. Other systems are available which ostensibly provide the same level of portability. However, we believe that our efforts have been paid back many times over in increased functionality, convenience, and control over the timing of bug-fixes and upgrades. Perhaps most important, the same libraries allow us to compile and run our code on a network of workstations, with all the conventional debugging and performance tools available. The end result is the ability to produce advanced software with less time debugging, and more time to produce scientific results.

We have also written a substantial amount of analysis and visualization software, which accounts for many more

lines of code than the parallel treecode itself. As the simulations have become larger, it has been necessary to write more and more of the analysis software so that it will run on a parallel machine. An example is the algorithm which identifies the location and size of individual halos in a cosmological simulation, which required us to implement a general parallel quicksort.

A final auxiliary software effort is the Self-Describing File format (SDF). We have found that the structure of our data files changes fairly frequently. In order to avoid the task of re-writing all our data input routines every time we compute another physical parameter, we have designed a file-format in which the structure of the binary data is described by an ascii header at the beginning of the file. Since the header is interpreted at runtime, old software doesn't even have to be recompiled to read new data files. This idea is, of course, not new, but to our knowledge, SDF is more expressive than other similarly motivated software (HDF and FITS in particular) and is also the only such software running in a parallel environment.

5 The future.

Scientific software is never complete. The problems of interest evolve and grow as new observations, new theories and new simulations emerge. After a few years of experience with a particular simulator, it is worthwhile to start afresh, creating a new code that incorporates the lessons laboriously learned from the old, as well as introducing new "physics", and new numerical techniques.

An entirely new treecode algorithm has been running for several months and is nearly ready for production The new *hashed oct-tree* (HOT) algorithm represents a major advance in style and sophistication. One of the limitations of the current production code is the complexity involved in determining locally essential data. The use of independent timestep integration methods which advance particles according to the local timescale is problematical with the old code. HOT can accommodate independent timestep integration schemes as well as an important new class of cell opening criteria [17] for which it is difficult to predict, *a priori*, which cells are required in a rectangular domain. HOT will also support $O(N)$ methods [4], which involve interactions between cells, in addition to the cell-body and body-body interactions which prevail in the $O(N \log N)$ methods we have used until now. The critical value of N at which $O(N)$ methods outperform $O(N \log N)$ methods is far from clear because the "constants" depend so strongly on both the specific problem and the most mundane of implementation details. However, by extending the error-estimates in [17] to describe the interactions present in $O(N)$ algorithms, we believe that superior performance can be achieved for realistic values of N.

The HOT method labels each possible cell with a key (a 64 bit integer derived from the spatial coordinates) and uses indirection through a hash table to locate data. Data stored in remote processors, as well as local data, can be requested through the hash-table inquiry routines. Minimizing latency, and tolerating the irreducible remainder by continuing with other work, are critical to the efficient functioning of the method. Thus, the inner loop of the algorithm is modified so that processing can continue while requests for non-local data are serviced in the "background". In this manner, locally essential data is acquired automatically, as it is needed, without a penalty from communication latency, and without requiring the ability to assemble a locally essential tree *a priori*.

Treecodes are by no means restricted to astrophysics. Many of the data generation and synthesis tasks associated with our simulation data can be expressed using hierarchical data structures, albeit not with gravitational interactions. Molecular dynamics simulations with charged or polar species (e.g., ions, water) are limited by the same $O(N^2)$ behavior as our gravitational simulations. Computational fluid dynamics using the *vortex method* requires solution of an N-body problem with a force law that is similar to that in gravity. Reaction-diffusion equations can be transformed into a computational problem with a similar structure. Even further afield are uses of spatial tree data structures in data compression, image processing, visualization and database searching. Our HOT implementation is far more flexible than the current production code. We believe it will not only be adaptable enough to satisfy our own needs for data analysis tasks, but will also enable us to collaborate with scientists in other disciplines to investigate broader uses of parallel treecodes.

Acknowledgments

We thank the CSCC and the CCSF for providing computational resources. JS wishes to acknowledge support from the Advanced Computing Division of the NSF, as well as the CRPC. MW wishes to acknowledge support from IGPP and AFOSR.

References

[1] R. W. Hockney and J. W. Eastwood, *Computer Simulation Using Particles*. New York: Mcgraw-Hill International, 1981.

[2] A. W. Appel, "An efficient program for many-body simulation," *SIAM J. Computing*, vol. 6, p. 85, 1985.

[3] J. Barnes and P. Hut, "A hierarchical O(NlogN) force-calculation algorithm," *Nature*, vol. 324, p. 446, 1986.

[4] L. Greengard and V. Rokhlin, "A fast algorithm for particle simulations," *J. Comp. Phys.*, vol. 73, pp. 325–348, 1987.

[5] S. B. Baden, *Run-time Partitioning of Scientific Continuum Calculations Running on Multiprocessors*. PhD thesis, U.C. Berkeley, 1987.

[6] J. K. Salmon, P. J. Quinn, and M. S. Warren, "Using parallel computers for very large N-body simulations: Shell formation using 180k particles," in *Proceedings of 1989 Heidelberg Conference on Dynamics and Interactions of Galaxies* (A. Toomre and R. Wielen, eds.), New York: Springer-Verlag, 1990.

[7] M. S. Warren, P. J. Quinn, J. K. Salmon, and W. H. Zurek, "Dark halos formed via dissipationless collapse: I. Shapes and alignment of angular momentum," *Ap. J.*, vol. 399, pp. 405–425, 1992.

[8] M. S. Warren and J. K. Salmon, "A parallel treecode for gravitational N-body simulations with up to 20 million particles," *baas*, vol. 23, no. 4, p. 1345, 1991.

[9] M. S. Warren, W. H. Zurek, P. J. Quinn, and J. K. Salmon, "The shape of the invisible halo: N-body simulations on parallel supercomputers," in *After the First Three Minutes Workshop Proceedings* (S. Holt, V. Trimble, and C. Bennett, eds.), New York: AIP Press, 1991.

[10] D. P. Fullagar, P. J. Quinn, and J. K. Salmon, "N-body simulations of a_4 isophote deviations in elliptical galaxies," in *Proceedings of ESO/EIPC Workshop on the Structure, Dynamics and Chemical Evolution of Early-Type Galaxies* (I. J. Danziger, ed.), (Munich), European Southern Observatory, 1992.

[11] C. Grillmair, *Dynamics of Globular Cluster Systems*. PhD thesis, Australia National University, 1992.

[12] C. L. Bennett *et al.*, "Preliminary separation of galactic and cosmic microwave emissions for the COBE DMR," COBE preprint 92-05, Goddard Space Flight Center, 1992.

[13] G. F. Smoot *et al.*, "Structure in the COBE DMR first year maps," COBE preprint 92-04, Goddard Space Flight Center, 1992.

[14] E. Bertschinger and J. M. Gelb, "Large cosmological N-body simulations," *Computers in Physics*, vol. 5, no. 2, p. 164, 1991.

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

[16] J. K. Salmon, "CUBIX: Programming hypercubes without programming hosts," in *Hypercube Multi-Processors 1987* (M. Heath, ed.), pp. 3–9, Philadelphia: SIAM, 1987.

[17] J. K. Salmon and M. S. Warren, "Skeletons from the treecode closet," *J. Comp. Phys.*, 1992. (in press).