PERSPECTIVES

FAST PARALLEL TREE CODES FOR GRAVITATIONAL AND FLUID DYNAMICAL N-BODY PROBLEMS

John K. Salmon
Physics Department, California Institute of Technology, Pasadena, California
Michael S. Warren[1]
Theoretical Astrophysics, Los Alamos National Laboratory, Los Alamos, New Mexico
Grégoire S. Winckelmans[2]
Graduate Aeronautical Laboratories, California Institute of Technology, Pasadena, California

We discuss two physical systems from different disciplines that make use of the same algorithmic and mathematical structures as a way of reducing the number of operations necessary to complete a realistic simulation. In the gravitational N-body problem, the acceleration of an object is given by the familiar Newtonian laws of motion and gravitation. The computational load is reduced by treating groups of bodies as single multipole sources rather than as individual bodies. In the simulation of incompressible flows, the flow may be modeled by the dynamics of a set of N interacting vortices. Vortices are vector objects in three dimensions, but their interactions are mathematically similar to that of gravitating masses. The multipole approximation can be used to greatly reduce the time needed to compute the interactions between vortices. Both types of simulations are carried out on the Intel Touchstone Delta, a parallel MIMD computer with 512 processors. Timings are reported for systems of up to 10 million bodies, and demonstrate that the implementation scales well on massively parallel systems. The majority of the code is common to both applications, which differ only in some of the "physics" modules. In particular, the code for parallel-tree construction and traversal is shared.

The International Journal of Supercomputer Applications, Volume 8, No. 2, Summer 1994, pp. 129–142

1. INTRODUCTION

Tree-based algorithms have had a major impact on the study of the evolution of gravitating systems as they provide a method of computing the mutual interactions of N bodies in much less than $O(N^2)$ time. Tree codes have been reported to scale as $O(N)$ or $O(N \log N)$, but the "big-O" notation can be misleading for practical values of N, where true performance is dominated by the constants that are discarded by the asymptotic analysis. Large-scale application of tree-based approximation methods has (to our knowledge) only occurred in astrophysics (Warren et al., 1992; Dubinski and Carlberg, 1991; Suginohara et al., 1991; Katz, Hernquist, and Weinberg, 1992) although preliminary work has been done on two-dimensional (Rokhlin, 1985; Pépin, 1990; Greengard, 1990; Engheta et al., 1992), and three-dimensional systems (Greengard and Rokhlin, 1989; Schmidt and Lee, 1991; Board et al., 1992; Ding et al., 1992) from other disciplines.

One roadblock to widespread acceptance of tree codes is that they are inherently difficult to program, especially for parallel machines. We report on an implementation of a tree code that is not specific to a particular problem domain. Although designed with astrophysical research firmly in mind, the code described here addresses example problems in vortex dynamics as well as in astrophysics. We report on its performance on the Intel Touchstone Delta system with up to 512 distributed-memory processors.

Two outcomes are possible when al-

1. Department of Physics, University of California, Santa Barbara.
2. New address: Department of Mechanical Engineering, University of Sherbrooke, Sherbrooke, Québec, Canada.

gorithmic advances drastically reduce the time and space required to solve a class of problems. The first is that the problems cease to be "supercomputer applications," and fall into the domain of workstations and personal computers. The second is that practitioners gradually advance the state-of-the-art in the underlying discipline, and much larger problems become the norm. In the latter case, supercomputers remain a critical component, and it is important to learn whether the new algorithm is well suited to supercomputer architectures, i.e., massively parallel systems. The second outcome has certainly been the case in astrophysics, where state-of-the-art simulations now evolve systems of 10^7–10^8 bodies. We expect it will also occur in other fields as tree-based methods become generally available.

2. THE GRAVITATIONAL *N*-BODY PROBLEM

2.1 Mathematics

While it is possible to describe tree methods in terms of broad generality and abstraction, we find it helpful to begin with a concrete example and develop the abstraction in stages. (Not coincidentally, this is also how our understanding of the problem evolved.) Newton himself would find the underlying mathematics of the gravitational N-body problem quite familiar. In modern notation, Newton's law of gravitation and his second law of motion are:

$$\frac{d^2}{dt^2}\,\mathbf{x}^s(t) = -\nabla\phi(\mathbf{x}^s(t),t) \qquad s = 1, \ldots, N, \tag{1}$$

$$\phi(\mathbf{x},t) = -\sum_q \frac{\mathrm{m}^q}{\|\mathbf{x} - \mathbf{x}^q(t)\|}. \tag{2}$$

We have chosen units in which the gravitational constant is unity. When evaluated at the position of one of the particles, $\mathbf{x}^s(t)$, the expression in Eq. (2) is clearly singular. The summation must be understood not to include the "self-interaction" of a mass-point with itself. Alternatively, it is possible to artificially "smooth" the Newtonian interaction, so that the gravitational potential between nearby bodies is bounded. This procedure also has the side effect of removing "collisions," something that is often desirable from a numerical or physical point of view (Hockney and Eastwood, 1981; Dyer and Ip, 1993). We can define a smoothed Green's function, $G_\sigma(\mathbf{x}) = (1/\sigma)G(\|\mathbf{x}\|/\sigma)$, and a smoothed potential:

$$\frac{d^2}{dt^2}\,\mathbf{x}^s(t) = -\nabla\phi_\sigma(\mathbf{x}^s(t),t) = -\sum_q \nabla G_\sigma(\mathbf{x}^s(t) - \mathbf{x}^q(t))\mathrm{m}^q. \tag{3}$$

Many choices are available for the smoothed Green's function. For the tests reported here, we use Plummer smoothing: $G(\rho) = 1/(\rho^2 + 1)^{1/2}$.

Equations (1) and (2) or (3) constitute a system of second-order ordinary differential equations. As such, it is not particularly difficult to integrate in time numerically. The computationally challenging part is the evaluation of the N right-hand sides, each of which is a sum of $N - 1$ terms. The gravitational force law is long range, i.e., the force law falls off slowly enough that contributions from distant objects cannot be assumed to vanish. Thus, it is not permissible simply to disregard contributions from bodies that are more distant than some prescribed cutoff.

We can turn again to Newton, at least for the first term in the solution. Let us imagine that the point $\mathbf{x}$ is well separated (in a sense defined below) from a spatially localized subset of the rest of the points, $\mathcal{S}$. Then the sum (at least over the bodies in $\mathcal{S}$) may be approximated by:

$$\sum_{q\in\mathcal{S}} \frac{\mathrm{m}^q}{\|\mathbf{x} - \mathbf{x}^q\|} = \frac{M_\mathcal{S}}{\|\mathbf{x} - \mathbf{x}_{cm}\|} + \frac{1}{2}\frac{\mathbf{Q}_{ij}(\mathbf{x} - \mathbf{x}_{cm})_i(\mathbf{x} - \mathbf{x}_{cm})_j}{\|\mathbf{x} - \mathbf{x}_{cm}\|^5} + \ldots \tag{4}$$

where $M_\mathcal{S}$ is the total mass in $\mathcal{S}$, $\mathbf{x}_{cm}$ is its center of mass, $\mathbf{Q}_{ij}$ is the quadrupole moment tensor of $\mathcal{S}$ about its center of mass,

$$\mathbf{Q}_{ij} = \sum_{q\in\mathcal{S}} \mathrm{m}^q(3(\mathbf{x}^q - \mathbf{x}_{cm})_i(\mathbf{x}^q - \mathbf{x}_{cm})_j - \delta_{ij}(\mathbf{x}^q - \mathbf{x}_{cm}) \cdot (\mathbf{x}^q - \mathbf{x}_{cm})) \tag{5}$$

and we use the Einstein summation convention implying summation over repeated vector/tensor indices. Equation (4) comprises the first terms of an expansion that describes the subset of points in terms of their mass, center of mass, quadrupole moments, octopole moments, etc. In general, the second term in the expansion contains the dipole moment of $\mathcal{S}$, but since the dipole moment vanishes about its center of mass, the dipole contribution to the force law vanishes by construction. The first term on the right-hand side of Eq. (4) approximates the contents of $\mathcal{S}$ as a point source at its center of mass. Subsequent terms are correct for the shape of the matter distribution within $\mathcal{S}$. In general, we can retain as many terms as desired in the expansion of the multipole moments of $\mathcal{S}$, designating the highest retained order as p, so $p = 1$ for the monopole approximation (since the dipole vanishes exactly, we may as well say that we've "retained" it), and $p = 2$ for the quadrupole approximation.

Equation (4) represents a major simplification, since its evaluation requires a fixed amount of time, regardless of how many points there are in $\mathcal{S}$. It is

analogous to the observation that if we wished to know the force exerted on an apple by the $\sim 10^{50}$ atoms in the Earth, we could approximate the Earth as a point mass located at its center of mass, and compute the force in a very small number of operations. It is by systematic application of Eq. (4) (or something mathematically equivalent) that all the fast N-body methods reduce the overall complexity from $O(N^2)$ to something much more tractable. Equation (4), of course, presupposes that the aggregate data, i.e., $M_{\mathcal{S}}$, $\mathbf{x}_{cm}$, etc., are known, so that any method that uses it must (a) compute the aggregate data efficiently, and (b) use each aggregate datum many times to amortize the cost of its computation.

We must also remember that Eq. (4) is an approximation. It must be used with care as it can introduce excessive errors into a calculation if used inappropriately. We have shown (Salmon and Warren, 1994) that the error introduced by using an approximation like Eq. (4) can be bounded as follows:

$$e_{\|\nabla\phi(\mathbf{x})\|} \leq \frac{1}{d^2} \frac{1}{\left(1 - \frac{b}{d}\right)^2} \times \left((p+2) \frac{B_{p+1}}{d^{p+1}} - (p+1) \frac{B_{p+2}}{d^{p+2}} \right) \quad (6)$$

where p is the highest order retained in the expansion of Eq. (4), $d = \|\mathbf{x} - \mathbf{x}_{cm}\|$, $b = \max_{q \in \mathcal{S}} \|\mathbf{x}^q - \mathbf{x}_{cm}\|$, and the moments B_n are defined as

$$B_n = \sum_{q \in \mathcal{S}} \|\mathbf{x}^q - \mathbf{x}_{cm}\|^n \mathrm{m}^q \quad (7)$$

Equation (6) is essentially a precise statement of the fact that the multipole approximation is more accurate when:

- The distance to the measurement point is large (large d).
- The sources are scattered over a small region (small b).
- High-order approximations are used (large p).
- Multipoles that have been neglected are small (small B_{p+1}).

Note that the multipole series does not converge at all for $d < b$, so this constitutes a precise statement of the condition that the point $\mathbf{x}$ must be "well-separated" from the points in $\mathcal{S}$. Furthermore, notice that increasing the order of the multipole approximation is just one way of improving the approximation. It is an open (and extremely interesting) question whether high-order approximations are cost effective, or whether it is simply better to restrict use of the approximation to larger values of d.

2.2 Data Structures

The approximation in Eq. (4) is only part of the story. A mechanism must be found to identify candidate subsets of bodies, compute their aggregate parameters (mass, center of mass, etc.), and selectively apply the multipole approximation to the evaluation of accelerations. Several different data structures have been proposed, including binary trees (Appel, 1985; Benz et al., 1990), 2^d-trees (Barnes and Hut, 1986) (where d is the dimensionality of the underlying space),[1] and multigrids of fixed depth (Greengard, 1987; Zhao, 1987). Tree structures are inherently adaptive, which means they can efficiently model systems that contain large density contrasts, while the multi-grid structures may enjoy some performance advantages because the fundamental objects are typically multi-dimensional arrays accessed in regular and predictable patterns, i.e., patterns suitable for efficient execution on vector and super-scalar processors. We have elected to work with adaptive oct-trees in three dimensions because the astrophysical problems that bred our initial interest in the topic are subject to extremely large and dynamic density contrasts, making adaptivity a necessary feature of any viable method. Some of the other problem domains where tree methods show promise may not require the adaptivity inherent in our code. It remains to be seen whether the overheads associated with adaptivity are significant vis-à-vis highly optimized nonadaptive codes.

An oct-tree is a partition of three-dimensional space into cubical volumes. Each cube has up to eight daughters, obtained by splitting the cube in half in each of the three Cartesian directions. Clearly, in d dimensions, the tree branches up to 2^d ways at each level. Quad-trees are far easier to illustrate on paper than oct-trees, so we use them in the figures. A quad-tree with 2,000 centrally concentrated bodies is shown in Figure 1. Notice that the tree is adaptive, so that in the center, where the density of bodies is high, the tree is more finely resolved. The adaptivity is achieved by building the quad-tree so that terminal cells contain exactly one body (Barnes and Hut, 1986). Thus, the depth of the tree is approximately logarithmic in terms of the local density of particles.

Each cell in an oct-tree has topological properties (parents, daughters, siblings), geometrical properties (spatial coordinates, size), and numerical properties (mass, center of mass, quadrupole moment, etc.). The programmer wishing to represent these properties is presented with several choices. The topological aspects of the tree can obviously be captured (at least on a uniprocessor) with pointers. Complications arise in parallel with regard to the meaning of off-processor pointers (for distributed-memory systems), or synchronization (for shared-memory systems), but these problems can be overcome (Salmon, 1990). Warren

1. We usually refer to quad-trees or oct-trees in two and three dimensions.

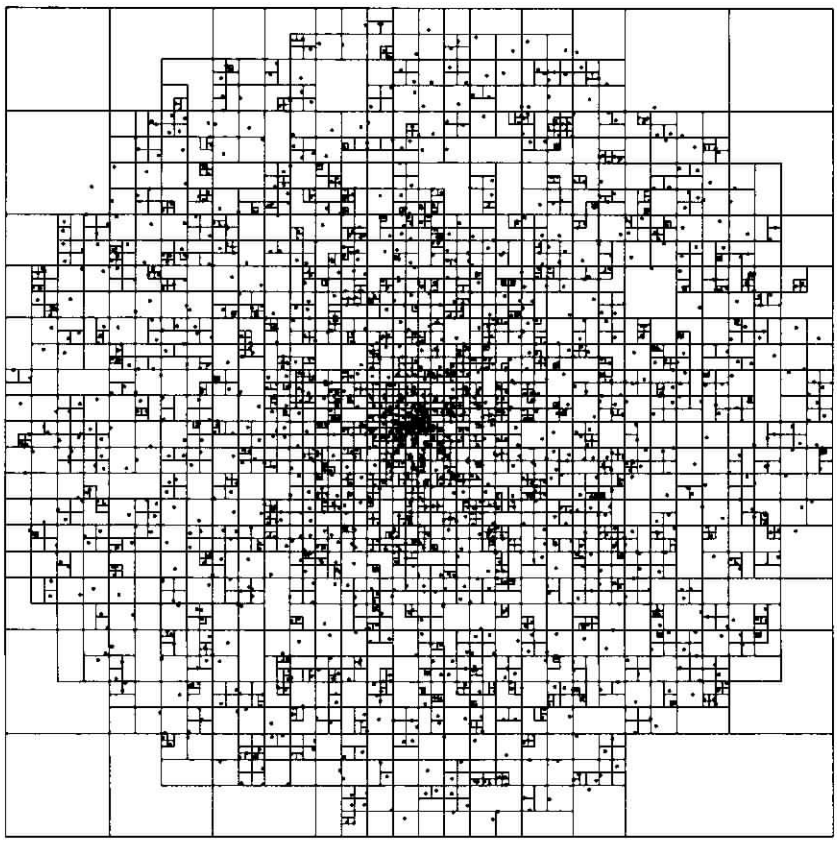

Fig. 1 A quad-tree with 2,000 centrally concentrated bodies The tree is more refined in regions of higher particle density because of the rule that a terminal cell can contain only one body.

and Salmon (1992) demonstrated large-scale parallelism when the data needed by each processor can be identified in advance. This method has been used in a number of astrophysical simulations (Salmon et al., 1990; Warren et al., 1992) that employed a much less rigorous error bound than that of Eq. (6).

Unfortunately, Eq. (6) does not allow us to determine in advance which data will be required by each cell. This is problematical for the methods of Salmon (1990), so a new approach to parallelism, based on accessing data on demand has been implemented (Warren and Salmon, 1993). The new method is built on the idea of assigning to every possible cell in the tree a unique multi-word key. With a 64-bit key it is possible to identify every cell in a oct-tree with 21 levels. This has proved adequate for highly clustered simulations with up to 10^7 bodies. The set of all *possible* keys is clearly much larger than the set of keys that will be present in any given simulation. This state of affairs suggests a hash table as an appropriate data structure for storing tree data.

Figure 2 shows schematically how the keys designate unique cells in the tree. The root has key 1. The daughters of any node are obtained by a binary left shift of the parent's key by d and then setting the low bits to a number between 0 and $2^d - 1$ to distinguish among the siblings. The parent of a cell is obtained by right shift of its key by d. The bodies in the simulation can be assigned unique keys as well, simply by finding the key corresponding to the smallest possible cell that contains the given body.

2.3 Control Structures

We have shown (at least schematically) how our tree will be represented in memory. We now turn to showing how to use the data to evaluate gravitational forces. For this, we must traverse the tree data structure, accumulating acceptable interactions as we proceed. The traversal is governed by a Multipole Acceptability Criterion (MAC) which tells us when Eq. (4) is sufficiently accurate to be used.

Many options are available for the MAC (Salmon and Warren, 1994). In this paper, we elect to bound the error introduced by each multipole approximation. Thus, we can use Eq. (6) di-

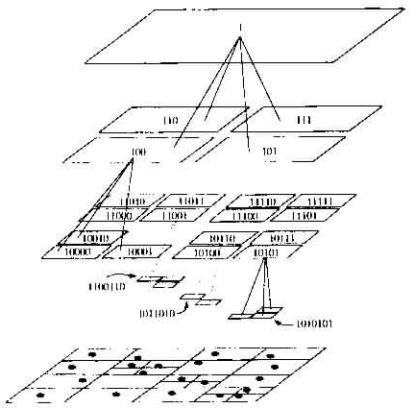

Fig. 2 A four-level quad-tree, expanded to show the relationship between parent cells and daughters The key values of each cell are shown in binary. Daughter keys are obtained from parents by a two-bit left-shift, followed by binary OR of a daughter-number in the range 00–11 (binary).

rectly and allow only interactions for which the right-hand side of Eq. (6) is less than some prescribed tolerance.

To use Eq. (6), we simply carry out a top-down traversal of the tree for each body independently, terminating the descent whenever Eq. (6) is satisfied by a cell. In those cases, Eq. (4) is evaluated to compute the influence of the contents of the cell on the body. If we reach a terminal node of the tree before Eq. (6) is satisfied, individual body-body interactions are computed. This strategy forms the basis of our vortex dynamics code (see Section 3).

For the gravitational problem, we have implemented an additional improvement by noting that particles that are near each other feel almost the same influence from distant objects (Appel, 1985; Rokhlin, 1985; Greengard, 1987). To extend our "apple" analogy further, now note that two neighboring apples feel almost the same gravitational field from the multitude of elementary particles constituting the Earth. Thus, if we wish to know the gravitational acceleration of two apples in the same tree, it may be sufficient to evaluate Eq. (4) once and use the same answer for both. In fact, this analogy again only captures the first (constant) term in a series expansion. This time, the expansion is a Taylor series for Eq. (4) around the point **x**. In order to pursue this line of reasoning, we need an error bound analogous to Eq. (6). We state here, without proof, the result that the combined error resulting from the dipole approximation and the linear term of a Taylor series expansion of the right-hand side of Eq. (4), at a distance Δ from **x** is bounded by:

$$e_{\|\nabla\phi(\mathbf{x}+\Delta)\|} \leq \frac{1}{d^2}\frac{1}{\left(1-\dfrac{\Delta+b}{d}\right)^2}\left(3\frac{\mathcal{B}_2}{d^2}-2\frac{B_3}{d^3}\right), \tag{8}$$

where

$$\mathcal{B}_{(2)} = B_2 + 2\Delta B_1 + \Delta^2 B_0. \tag{9}$$

Use of Taylor series to reduce the number of evaluations of Eq. (4) introduces a new class of interactions into the problem. Formerly, Eq. (4) was all that was required to evaluate the interactions between bodies and the aggregate data stored in cells. Now it is also necessary to evaluate interactions between *source* cells, i.e., the aggregate data describing the field sources, mass, center of mass, etc., and the *sink* cells, i.e., the coefficients of a Taylor series expansion around some arbitrary point. In addition, we must compute *translations* of the origin of a Taylor series expansion from the center of a parent sink cell to the centers of its daughters.

We can compute the field at every body position by looping over all of the body positions in order of increasing key. For each body, we find the nearest common ancestor with the previous body. Any Taylor expansions above that common ancestor remain valid for the new body, while those in the intervening levels must be computed by *translating* the Taylor expansion of the parent and accumulating any cell-cell interactions that satisfy the extended cell-cell MAC (based on Eq. (8)). It is crucial for bodies to be sorted in hash-key order as that guarantees a high correlation between the spatial positions of successive bodies, so that only a small number (one, on average) of Taylor expansions will need to be recomputed for each new body. Thus, use of the Taylor series expansion reduces the asymptotic order of the method to $O(N)$, but this means little unless the constants of proportionality are also compared. The evaluation of the coefficients of the Taylor series can be much more time consuming than the evaluation of Eq. (4) alone, so the benefit of fewer interactions is partially cancelled out by the greater cost of those interactions. Preliminary results on a one-million body system indicate that the total number of floating-point operations is about a factor of four lower when using cell-cell interactions and Eq. (8) compared with use of body-cell interactions (Eq. (4)) and Eq. (6).

2.4 Parallelism

Two basic issues arise in parallelizing many scientific algorithms for distributed-memory machines: decomposition and data acquisition. Decomposition refers to the strategy employed to partition the problem among the available processors. Data acquisition refers to the need to communicate between processors so that they have the data they need to carry out the necessary computations on their subset of the data. Decomposition is concerned with load balance, i.e., arranging that all the processors finish at approximately the same time, and with minimizing the volume of communication. Data acquisition is often concerned with minimizing or hiding the high latency associated with every interprocessor message. This is achieved by overlapping communication and computation when the hardware supports it, and/or by buffering messages into large blocks that can be sent with the latency overhead of only a single message.

We have already seen that it is desirable to sort the bodies in our simulation into a sequence based on the hash-table key. We return to the concept of a key-sorted list of bodies to describe our parallel decomposition. Imagine that the bodies are already sorted in increasing hash-key order. We simply break the list into segments and assign each segment to a processor. A processor becomes responsible for computing the forces on a group of bodies that forms a contiguous segment of the sorted list of bodies. Figure 3 shows a path that traces out values of increasing key in a quad-tree with depth three. The decomposition is achieved by cutting the curve into P

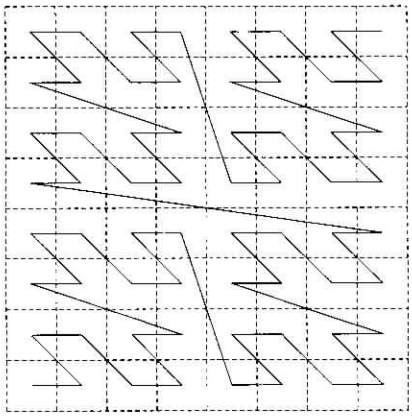

Fig. 3 A curve that traces increasing values of the hash-function key in a quad-tree of depth three The decomposition is obtained by locating particles on this curve (more precisely, on the analogous curve that fills the lowest level of the tree), and assigning contiguous segments to processors.

(number of processors) segments of equal cost.

By dividing the path into equal-cost pieces, we are allowing that some bodies are more expensive than others. The cost of each body is a measure of how much cpu time is required to compute the forces on it. In a highly clustered, adaptive tree, the ratio of the most expensive to the least expensive body can be as high as 20, so it is important to balance actual cost, rather than just particle number. We determine the decomposition-cost of each body empirically by recording the number of interactions it required on the previous timestep, assigning every body equal cost on the first timestep. This can lead to considerable load imbalance on the first timestep, but the effect is negligible over the course of a simulation that typically requires hundreds or thousands of timesteps.

This decomposition has the advantage that it can be generated with a general-purpose parallel sort routine, and also that it leads to spatial locality in the decomposition. The latter property reduces the amount of interprocessor communication traffic, as spatially nearby bodies tend to require the same off-processor information. In contrast to the decomposition used by Salmon (1990), boundaries between processors correspond to divisions within the tree-data structure. Notice also that from one timestep to the next, the bodies do not move significantly, so that on every timestep but the first, the sorting subroutine is presented with almost-sorted input.

The second issue in parallelism is the acquisition of off-processor data. One strategy, discussed by Salmon (1990), is to acquire all the necessary off-processor data in a communication phase prior to the computation. This has the advantage of leaving the main loop of the computation essentially untouched, allowing for re-use of highly optimized sequential code. In this technique it must also be possible to determine *a priori* which data will be needed by which processors. Unfortunately, when Eq. (6) is used for the MAC, it is not possible to predetermine which data will be necessary. Thus, we adopt a new strategy in which off-processor data is acquired on demand.

This new scheme requires a small modification of the hash-table data structure used to represent the tree. In parallel, the hash-table lookup functions can return in one of three possible ways: they can find the requested item, they can report that the requested item does not exist, or they can report that the requested item exists in another processor's memory. In the last case, it is left up to the caller how to proceed. The simplest approach is to simply execute a remote procedure call in order to retrieve the remote data. This is unacceptably slow, as it entails several times the full message latency for every remote data access. An alternative is to enqueue a request to be dispatched at a later time (after many more requests have been enqueued), and to proceed with other branches of the tree, or with other particles. This greatly reduces the total number of messages that have to be sent, and, hence, overhead due to message latency. In addition, in this approach it is possible to almost totally overlap communication with calculation, since if the hardware supports simultaneous communication and calculation, the code can easily take advantage of it. On the other hand, it is necessary to make substantial changes to the critical inner loop of the tree traversal routine in order to allow for deferred processing of off-processor cells and simultaneous processing of several bodies.

2.5 Performance

We report here some results obtained with the gravitational code on the Intel Touchstone Delta system. Although the Delta has 576 processors (both 80386 and i860), a maximum of 512 of the 40-MHz i860's may be assigned to a single job at any one time (the others are responsible for various system tasks like managing peripheral devices, user logins, etc.). Each processor has 16 Mb of memory and is connected to four neighbors through a two-dimensional mesh routing network. Interprocessor communication is accomplished by means of explicit system calls embedded in the application's C or FORTRAN code. There is no hardware support for shared memory. The code is written entirely in ANSI C, and has been ported to several other parallel platforms, including the CM-5, the SP-1, workstation networks, the Intel Paragon, and the Ncube-2. Notably, the code also compiles and runs with no modifications whatsoever on sequential platforms that supply an ANSI C compiler.

We report results that use a monopole approximation for the force law and a first-order Taylor series for translation. Thus, the first neglected

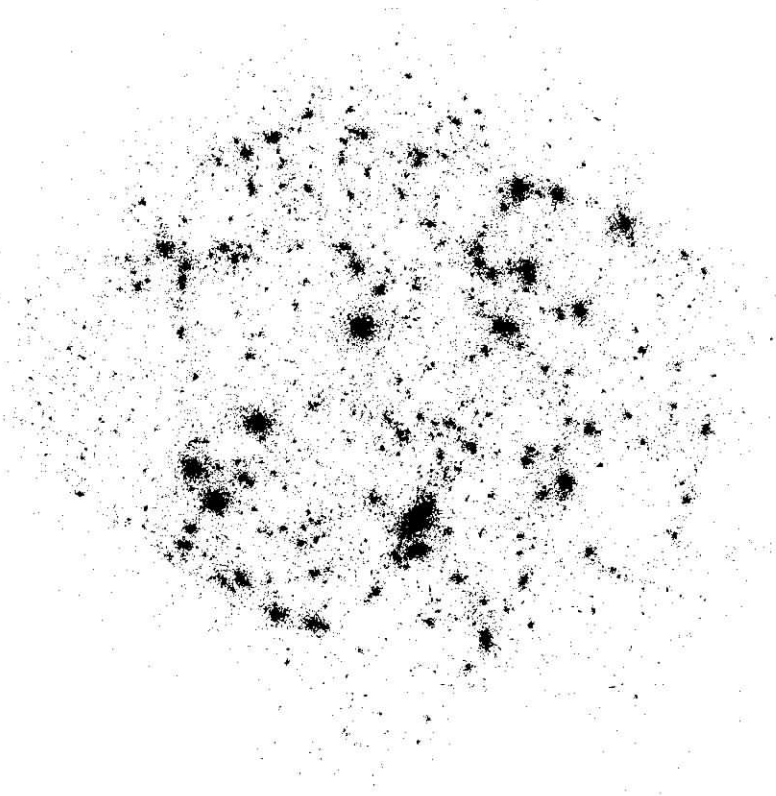

Fig. 4 Points are located at the (x,y) projection of the three-dimensional positions of the bodies in our example N-body problem. The figure shows 86,995 bodies chosen at random from the full data set. The region shown is 10 Mpc across.

error term is always inversely proportional to the square of the distance between the source and the observation point. Errors in the acceleration are analytically bounded, using Eq. (8), to be less than 2% of the force at the edge of the spherical domain. In practice, the errors are typically much less than 2%.

The particle distribution is taken from a series of simulations designed to simulate the formation of large-scale structure in the universe.[2] We chose a representative distribution from late in the evolution that displays significant clustering. The density contrast between the densest and least dense regions in the simulation is approximately 10^6. The particles represent the so-called "dark matter" which is believed to dominate the mass of the universe. The region simulated is a sphere 10 Mpc in diameter, containing 1,099,135 bodies. Each body has a mass of approximately 3.3×10^7 solar masses, which is appreciably less than the mass of a typical galaxy (10^{11}–10^{13} solar masses). To generate data sets with fewer bodies we simply chose bodies at random from the full data set. While this procedure may be questionable from a physical point of view, it ensures that the scaling behavior (as a function of N) will not be contaminated by differences arising from different spatial distributions of bodies. Figure 4 shows approximately 87,000 of the bodies in our data set.

Figure 5 shows timings for data sets with N = 5,000, 10,000, 20,000, 50,000, 100,000, 200,000, 500,000 and 1,099,135 bodies on 1, 2, 4, . . . , 512 processor subsets of the Delta. The times reported are the total time per timestep (force evaluation, velocity update, position update), averaged over 2 to 5 timesteps (more for less time-consuming runs). We do not count the first timestep because it suffers from an anomalous load imbalance which is corrected in later steps after the "cost" of each particle is known. The variation between timesteps is typically a few percent, except for the first, which typically takes about twice as long as the others. The machine is not dedicated during these benchmarks, so variations may be due

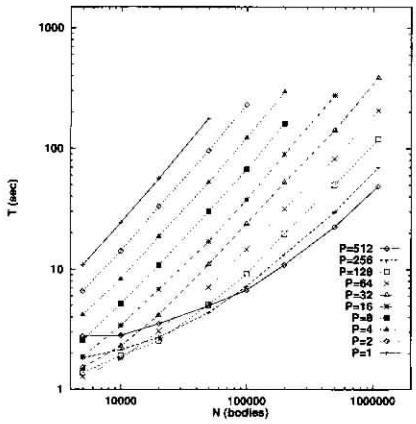

Fig. 5 Running time, in seconds, for each iteration of the gravitational N-body problem Times are reported for processor number P = 1, 2, . . . , 512. They are computed by averaging 2–5 successive iterations. In all cases, the initial timestep is not included because it suffers from load imbalance and I/O startup.

2. It is timestep 540 of model 128m.n-1b in Warren et al. (1992).

to the activity of other users or to inherent variations from one timestep to the next.

The fact that the curves are parallel and well separated for large N indicates that adding processors really does reduce the execution time. In order to remove the dominant trends in the data, we plot in Figure 6 the quantity TP/N against N. In this figure, "perfect" parallelism is represented by all curves lying on top of the extrapolation of the $P = 1$ curve. It is clear that the $P = 512$ curve is still approaching the others, i.e., we have not yet reached the large-N limit, even with over one million bodies. In addition, Figure 6 shows that the scaling of T with N is slightly super-linear for the range of N studied. In fact, it is close to $T \propto N^{1.25}$, but it is very difficult to distinguish between slightly different functional forms of the asymptotic scaling behavior with only four $P = 1$ data points.

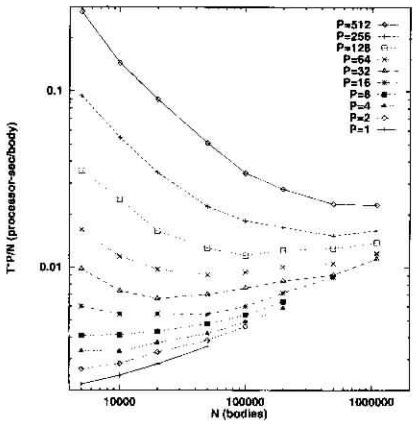

Fig. 6 The same data as Fig. 5, but the abscissa is *TP/N* The vertical distance from the (extrapolation of the) $P = 1$ curve to the other curves is a good indicator of parallel overhead. In addition, these curves indicate that the time per timestep scales slightly super-linearly with N. It is clear that parallel overhead has still not reached its "large-N limit" with 1 million bodies on 512 processors.

We have also run a series of benchmarks with a uniform, spherical distribution of points (not shown). In Figure 7, we plot TP/N against N for up to 10 million uniformly distributed bodies. The force evaluations are 3–5 times faster in the case of a uniform distribution of particles, compared with the highly clustered, evolved astrophysical data. In addition, the N-dependence appears to be very close to $T \propto N$. A detailed analysis of the dependence of the time-per-timestep on the particle distribution will be presented elsewhere.

3. THE VORTEX PARTICLE *N*-BODY PROBLEM

3.1 Mathematics

The vorticity equation ($\boldsymbol{\omega} = \nabla \times \mathbf{u}$, and hence $\nabla \cdot \boldsymbol{\omega} = 0$) for an incompressible fluid ($\nabla \cdot \mathbf{u} = 0$) is obtained by taking the curl of the momentum equation:

$$\frac{D\omega}{Dt} = (\nabla \mathbf{u}) \cdot \boldsymbol{\omega} + \nu \nabla^2 \boldsymbol{\omega}, \quad (10)$$

where $Df/Dt = \partial f/\partial t + (\mathbf{u} \cdot \nabla) f$ is the Lagrangian derivative and ν is the kinematic viscosity. The vorticity equation is thus a nonlinear transport equation that can be solved using a particle method. In the regularized version of the method (Rehbach, 1977; Chorin, 1980, 1981; Saffman, 1980; Saffman and Meiron, 1986; Leonard, 1985; Beale and Majda, 1982a, 1982b; Novikov, 1983; Anderson and Greengard, 1985; Mosher, 1985; Beale, 1986; Choquin, 1987; Cottet, 1988; Choquin and Cottet, 1988; Chua et al., 1988; Winckelmans, 1989; Winckelmans and Leonard, 1988, 1989, 1993) the particle representation of the vorticity field is taken as:

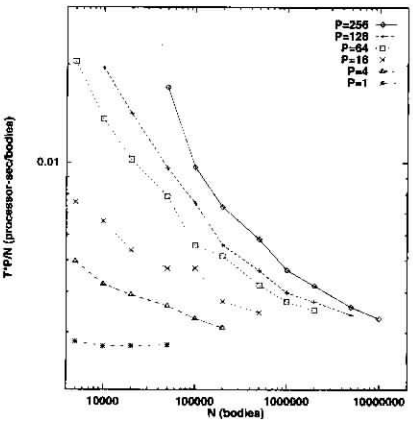

Fig. 7 Timing data for a uniform, spherical distribution of bodies plotted with an abscissa of *TP/N*. The vertical distance from the (extrapolation of the) $P = 1$ curve to the other curves is a good indicator of parallel overhead. For uniform data, the N dependence is very close to linear, and parallel overhead is less than 50% for 10 million bodies on 256 processors.

$$\tilde{\boldsymbol{\omega}}_\sigma(\mathbf{x},t) = \sum_q \zeta_\sigma(\mathbf{x} - \mathbf{x}^q(t)) \left(\boldsymbol{\omega}^q(t) \frac{\mathrm{vol}^q}{4\pi}\right) = \sum_q \zeta_\sigma(\mathbf{x} - \mathbf{x}^q(t)) \boldsymbol{\gamma}^q(t). \quad (11)$$

where ζ_σ is a radially symmetric regularization function and σ is a smoothing radius (i.e., a core size): $\zeta_\sigma(\mathbf{x}) = (1/\sigma^3)\, \zeta(\|\mathbf{x}\|/\sigma)$ with the normalization $\int_0^\infty \zeta(\rho)\rho^2\, d\rho = 1$. Notice that $\tilde{\boldsymbol{\omega}}_\sigma$ does not constitute a generally divergence-free "basis."

The velocity field is computed from the particle representation of the vorticity field as the curl of a vector stream function, $\mathbf{u}_\sigma = \nabla \times \tilde{\boldsymbol{\psi}}_\sigma$ (hence $\nabla \cdot \mathbf{u}_\sigma = 0$). The vector stream function thus satisfies $\nabla^2 \tilde{\boldsymbol{\psi}}_\sigma(\mathbf{x},t) = -\tilde{\boldsymbol{\omega}}_\sigma(\mathbf{x},t)$. Defining $G_\sigma(\mathbf{x}) = (1/\sigma)\, G(\|\mathbf{x}\|/\sigma)$ with $G(\rho)$ such that

$$-\zeta(\rho) = \nabla^2 G(\rho) = \frac{1}{\rho^2} \frac{d}{d\rho}\left(\rho^2 \frac{dG}{d\rho}\right), \quad (12)$$

gives us

$$\tilde{\boldsymbol{\psi}}_\sigma(\mathbf{x},t) = \sum_q G_\sigma(\mathbf{x} - \mathbf{x}^q(t))\boldsymbol{\gamma}^q(t) \tag{13}$$

$$\mathbf{u}_\sigma(\mathbf{x},t) = \sum_q (\nabla G_\sigma(\mathbf{x} - \mathbf{x}^q(t))) \times \boldsymbol{\gamma}^q(t), \tag{14}$$

Gaussian smoothing is used:

$$\zeta(\rho) = \left(\frac{2}{\pi}\right)^{1/2} e^{-\rho^2/2}, \tag{15}$$

$$G(\rho) = \frac{1}{\rho}\,\mathrm{erf}\left(\frac{\rho}{\sqrt{2}}\right). \tag{16}$$

It leads to a second-order numerical method, provided the core-overlapping condition remains satisfied (i.e., $\sigma/h \geq 1$ where h is a typical spacing between the particles). We approximate $\mathrm{erf}\,(\rho/\sqrt{2}) \approx 1$ for $\rho > 4$, i.e., for $\|\mathbf{x}\| > 4\sigma$, we have $G_\sigma(\mathbf{x}) = 1/\|\mathbf{x}\|$.

In the inviscid case, the evolution equations for the particle position and strength vector are taken as

$$\frac{d}{dt}\mathbf{x}^s(t) = \mathbf{u}_\sigma(\mathbf{x}^s(t),t), \tag{17}$$

$$\frac{d}{dt}\boldsymbol{\gamma}^s(t) = (\nabla \mathbf{u}_\sigma(\mathbf{x}^s(t),t)) \cdot \boldsymbol{\gamma}^s(t). \tag{18}$$

Evaluation of the velocity field induced by a system of vortex particles is, thus, very similar to evaluation of the acceleration field induced by a system of point masses in gravitation. Equation (14) is structurally almost identical to Eq. (3) except for having to replace the scalar particle mass m^q by the vector vorticity strength, $\boldsymbol{\gamma}^q$, and scalar multiplication by a vector cross-product. Just as in the gravitational case, it is possible to approximate the summation in Eq. (14) by an expression similar to Eq. (4) involving the multipole moments of the vorticity distribution. The asymptotic behavior of the Green's function, G_σ, is even the same, which allows us to re-use much of the machinery involving error bounds given by Salmon and Warren (1994). A vortex particle code is, however, more costly than a gravitational code since (1) the particle strength and potential fields are vectors rather than scalars, so the potential evaluation is immediately three times as costly, and (2) both the first and second derivatives of the vector streamfunction must be evaluated in order to obtain both the velocity vector and the velocity gradient tensor, which appears in the right-hand side of Eq. (18).

Other difficulties arise from the interpretation of the vortex particles as representing a continuum. Hence, a smoothing function that leads to convergence, e.g., Gaussian smoothing, must be used; note that Gaussian smoothing is considerably more costly than the Plummer smoothing, used in the gravitational code. In vortex simulations, we also have to ensure that the smoothed vortex particles continue to overlap for the duration of the simulation. Hence particle "redistribution" (from a deformed set of particles onto a new set of regularly spaced particles) may become necessary in long-time computations (Koumoutsakos, 1993; Koumoutsakos and Leonard, 1992). Finally, the particle field, $\tilde{\boldsymbol{\omega}}_\sigma$, is not guaranteed to remain a good representation of the divergence-free field, $\boldsymbol{\omega}_\sigma = \nabla \times \mathbf{u}_\sigma$, for all times. Hence, in long-time computations, it may be necessary to "relax" the particle weights so that $\tilde{\boldsymbol{\omega}}_\sigma$ remains a good representation of $\boldsymbol{\omega}_\sigma$ (Winckelmans, 1989; Winckelmans and Leonard, 1993; Pedrizzetti, 1992). These considerations apply to any vortex method and are not significantly affected, one way or the other, by use of a tree code to carry out the field summations.

When using multipole expansions up to $p = 2$ (monopole + dipole + quadrupole), estimates for the error on $\tilde{\boldsymbol{\psi}}$ and $\mathbf{u} = \nabla \times \tilde{\boldsymbol{\psi}}$ at an evaluation point $\mathbf{x}$ a distance d from the center $\mathbf{x}_c$ of a multipole expansion are obtained as:

$$e_{\|\tilde{\psi}\|} \leq \frac{1}{d}\frac{1}{\left(1 - \frac{b}{d}\right)}\frac{B_3}{d^3} \leq \frac{1}{d}\frac{1}{\left(1 - \frac{b}{d}\right)}\frac{bB_2}{d^3},$$

$$e_{\|\mathbf{u}\|} \leq \frac{1}{d^2}\frac{1}{\left(1 - \frac{b}{d}\right)^2}\left[4\frac{B_3}{d^3} - 3\frac{B_4}{d^4}\right] \tag{19}$$

$$\leq \frac{1}{d^2}\frac{1}{\left(1 - \frac{b}{d}\right)^2}\left[4\frac{bB_2}{d^3} - 3\frac{B_2^2}{B_0 d^4}\right], \tag{20}$$

where $b = \|\mathbf{x}^q - \mathbf{x}_c\|_{\max}$, B_0 and B_2 are box properties:

$$B_0 = \sum_q \|\boldsymbol{\gamma}^q\|, \tag{21}$$

$$B_2 = \sum_q \|\mathbf{x}^q - \mathbf{x}_c\|^2 \|\boldsymbol{\gamma}^q\|. \tag{22}$$

We choose

$$\mathbf{x}_c = \frac{\Sigma_q \|\boldsymbol{\gamma}^q\|\,\mathbf{x}^q}{B_0}, \tag{23}$$

the centroid of the absolute value of the particle strengths, as the center of our multipole expansion. This choice analytically minimizes B_2, and hence minimizes the bound on the error introduced by the multipole expansion. Notice that the dipole term of this multipole expansion does not vanish. The dipole moment vanishes only in the gravitational case because the masses are positive-definite scalars, and the centroid of the absolute value of the masses is identical with their center of mass. The dipole contribution does not vanish in the case of elec-

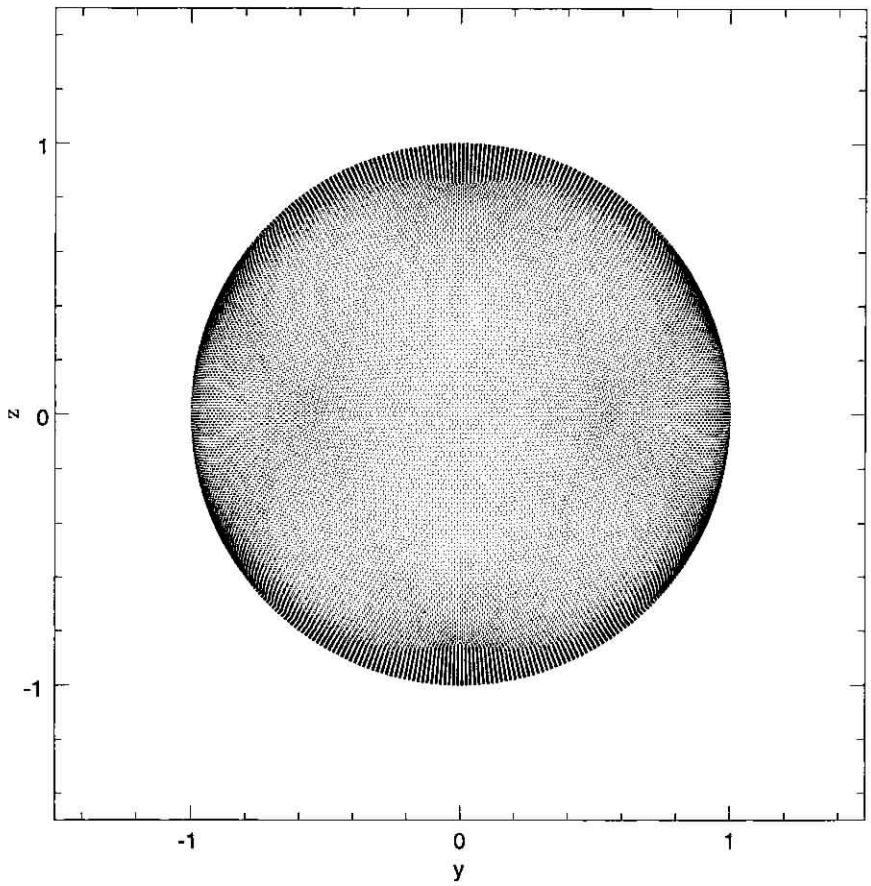

Fig. 8 The positions of 81,920 vortex particles initially on the surface of a unit sphere Each of the particles carries a vector strength μA, with $\mu = 3/8\pi \sin(\theta)\hat{e}_\varphi$ and A the area of the projected surface element associated with that particle. The vectors are not shown for clarity. This simulation is slightly different from those for which timings are presented because it uses a core size $\sigma = 0.05$.

trostatics either, where each particle has a scalar electric charge that may be of either sign.

3.2 Performance

We carried out a series of timings for a problem representing the evolution of an initially spherical vorticity distribution. Figure 8 shows the initial positions of vortex particles representing a surface vorticity of sheet strength,

$$\boldsymbol{\mu} = \frac{3}{8\pi} \sin(\theta)\hat{e}_\varphi, \qquad (24)$$

which is the solution to the problem of flow past a sphere with unit free-stream velocity in the z-direction. The problem is discretized by recursively splitting the faces of an icosahedron into equilateral triangles and then projecting them onto a unit sphere. The strength of each vortex particle is taken to be $\boldsymbol{\gamma} = \boldsymbol{\mu}A$, where A is the area of the projected triangle, and $\boldsymbol{\mu}$ is the value of Eq. (24) at its centroid. By terminating the recursive splitting at different levels, we obtain the discretizations shown in Table 1. The core size, σ, is taken to be equal to the linear size of the unprojected triangles. Figure 9 shows the 81,920-vortex system evolved to $t = 2.50$ (after 100 timesteps). Note that although we depict the particles as points, they actually carry a three-component strength vector $\boldsymbol{\gamma}$. Representing the particles as vectors merely clutters the figure. The simulations were done using a "sum" error tolerance (see Salmon and Warren, 1994) based on Eq. (20), which guarantees that the total error in the velocity is less than 0.001.

Timings are shown in Figure 10 for various $P = 1, 2, 4, \ldots 512$, and for the discretizations shown in Table 1. The timings correspond to the total wallclock time per iteration, i.e., the time spent in parallel decomposition, computing the vector streamfunction $\boldsymbol{\psi}$, the velocity $\mathbf{u}$, and the gradient of $\mathbf{u}$ at each particle position, as well as updating the particle positions and strengths according to Eqs. (17) and (18). As with the gravitational code, we report an average over 2–5 iterations, and we do not include the first timestep because of its anomalous parallel load imbalance. Comparison with the gravitational timings confirms the expectation that the vortex code is somewhat more costly than the gravitational code.

The same data is replotted with an abscissa of TP/N in Figure 11 to better illustrate the dominant trends. Here it is clear that one million bodies clearly

Table 1
Parameters for the Series of Vortex Particle Method Simulations

Level	σ	N
4	0.0657	5,120
5	0.0329	20,480
6	0.0164	81,920
7	0.00821	327,680
8	0.00411	1,310,720

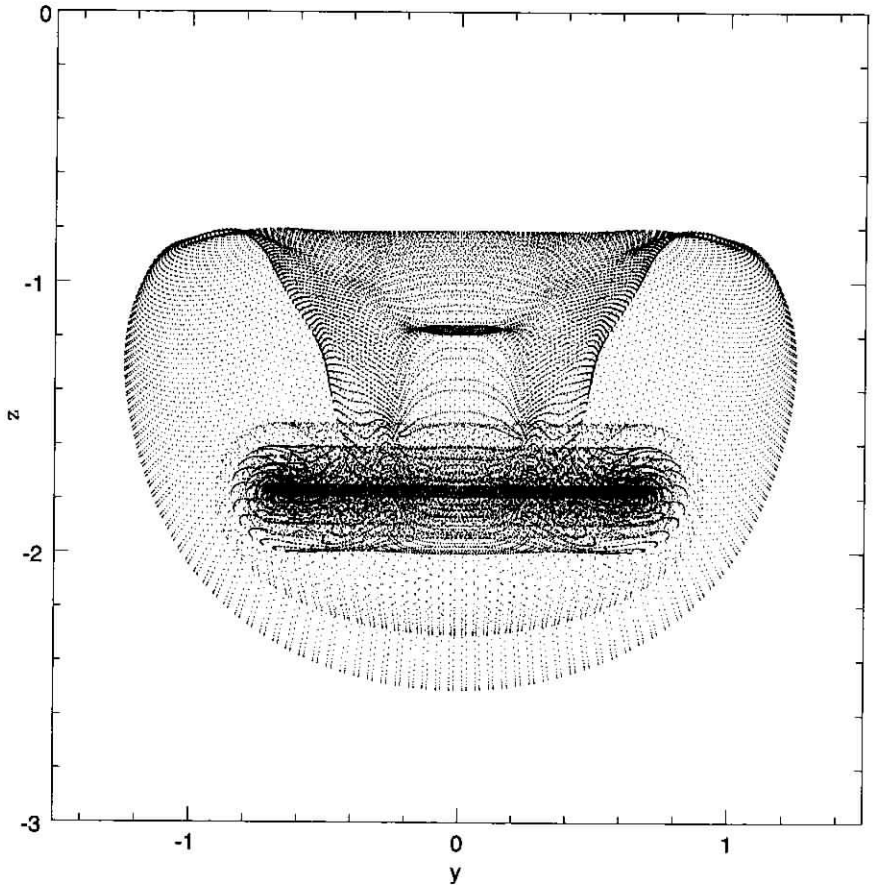

Fig. 9 The vortex simulation of Fig. 8, evolved through 100 timesteps to $t = 2.5$.

is in the "large-N" limit, and that the parallel overhead (obtained by measuring the difference between the $P = 512$ curve and the extrapolation of the $P = 1$ curve) is in the neighborhood of 20%. Thus, although the vortex simulation is overall somewhat slower than the gravitational simulation, it makes more efficient use of the parallel hardware (a fact that is of small consolation to the user with a limited computational budget). Figure 11 also demonstrates that the scaling behavior with N is again slightly super-linear. This time the exponent is approximately $T \propto N^{1.2}$.

4. CONCLUSION

The two methods described here demonstrate that tree codes are versatile and scale well on large parallel computers. Despite the complexity of the algorithms involved, we have been able to use essentially the same code to solve problems in vortex dynamics and astrophysics. The code, including all auxiliary software for such tasks as random number generation, fast approximate square root, a primitive debugging facility, flexible timers and counters, etc., as well as the essential elements (parallel quicksort, multi-

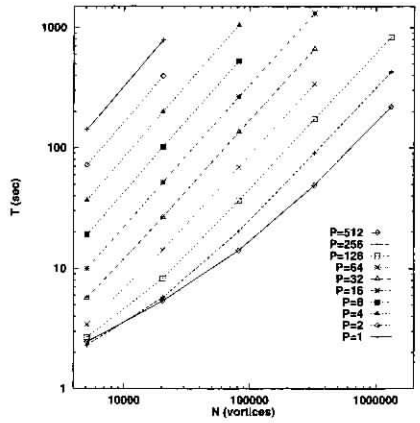

Fig. 10 Time per timestep for the vortex method versus number of vortices, averaged over several timesteps.

word key and hash-table manipulation, tree construction, and traversal and field evaluation) consists of about 9,000 lines of C source code and header files. Of this, about 7,000 lines are in a common library that is completely shared by the two applications (in fact, much of the library is not even

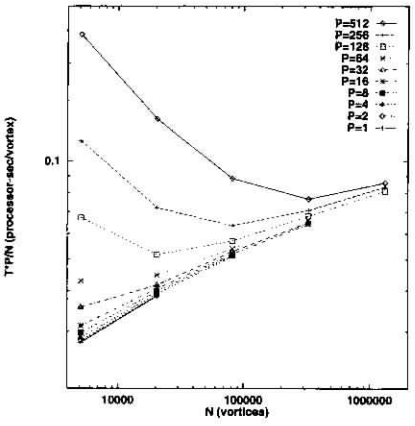

Fig. 11 The same data as Figure 10, but plotted with an abscissa of *TP/N* The parallel overhead can be estimated by measuring the difference between a $P \neq 1$ curve and the (extrapolation of the) $P = 1$ curve. The large N dependence is again slightly super-linear over the range of N studied.

specific to tree codes). Of the remaining 2,000 lines, about 700 are identical in the two applications. We are pursuing further abstractions that will allow us to separate these into the common library as well.

It is important to note that the programs and libraries described here are portable to other parallel supercomputers. We have run the code on Intel Paragon and iPSC/860 systems, an IBM SP-1, a CM-5, an Ncube-2, and on networks of workstations, as well as the "degenerate" parallel case of a single uniprocessor workstation. All that is required to support a new system is the creation of an appropriate Makefile to define such things as the C compiler, linker, etc., and the creation of a single system-dependent C source file. The system-dependent file maps a very small number of primitive OS requests (e.g., What processor number am I?) into system calls appropriate for the target architecture. The Delta version of this file is 400 lines in length and con-sists primarily of name-translations from, e.g., our internal name `Procnum ()` to the OS-specific name `myproc ()`. Porting this file to a new platform typically requires a few hours, most of which is spent searching the target system's manuals for relevant system calls.

Portability and versatility are extremely important attributes for the parallel tree code. We are actively pursuing other application areas where tree codes show great promise. The electrostatic interaction in molecular dynamics is long range and has the same functional form as Newtonian gravity. In some circumstances it can dominate the time required to carry out a simulation. In fact, the perceived cost of calculating electrostatic interactions may influence the choice of problems or the approximations that are studied. Several authors have used tree codes to address this problem, but, to our knowledge, none have used parallel architectures and none have used error-bound criteria like Eq. (6).

Integral equations in potential theory were historically the first application of tree codes (Rokhlin, 1985). Related problems give rise to the largest problems currently addressed with $O(N^3)$-dense linear solvers (Edelman, 1993). We are in the process of completing a tree-code implementation of a boundary integral equation solver which will be incorporated into our vortex dynamics code to account for the presence of bluff bodies in the flow. The resulting program will use tree codes in two separate contexts: to compute the interactions between vortices, and to compute the interactions between surface panels as part of an iterative solver to satisfy boundary conditions on the surfaces. The interactions are different in the two contexts, as are the spatial distribution of sources and the error criteria, but the parallel tree code library provides much of the necessary machinery for both contexts. Preliminary results indicate that one million–panel systems can be solved on the Delta in a few minutes. Such problems would be completely intractable using traditional solvers.

ACKNOWLEDGMENT

This research was performed in part using the Intel Touchstone Delta system operated by Caltech on behalf of the Concurrent Supercomputing Consortium. Access to this facility was provided by Caltech and LANL. JKS and GSW were supported by the National Science Foundation Center for Research in Parallel Computation. GSW was also supported by Office of Naval Research, Grant No. N00014-92-J-1072. MSW and JKS were supported in part by a grant from the National Aeronautics and Space Administration under the HPCC program.

BIOGRAPHIES

John Salmon graduated from MIT in 1981 with an S.B. in Physics and EECS. He has an M.S. (1983) in Physics from University of California, Berkeley and a Ph.D. (1990) in Physics from the California Institute of Technology. His research is in the application of high-performance parallel computers and "fast" methods to problems in astrophysics and other disciplines. He is currently a staff member with the Caltech Concurrent Supercomputing Facility.

Michael S. Warren received his B.S. in Physics and Engineering from the California Institute of Technology in 1988, and a Ph.D. in Physics from the University of California, Santa Barbara in 1994. He has been active in the development of parallel tree codes since 1987. He is currently applying fast parallel tree codes to current problems in cosmology and astrophysical hydrodynamics as a technical staff member in the Theoretical Astrophysics group at Los Alamos National Laboratory.

Grégoire S. Winckelmans earned a Diplôme d'Ingénieur Civil Mécanicien (Mechanical Engineer) from the Université Catholique de Louvain à Louvain-la-Neuve in 1983, a Postgraduate Diploma from the von Karman Institute for Fluid Dynamics in 1984 and an M.S. (1985) and a Ph.D. (1989) in Aeronautics from the California Institute of Technology. He was also a postdoctoral Research Fellow there. He is currently an Assistant Professor of Mechanical Engineering at the University of Sherbrooke in Québec. His research involves the development of fast par-

ticle vortex methods and fast boundary-element methods for problems in computational fluid dynamics, and of other particle methods for problems in computational combustion.

SUBJECT AREA EDITOR

James Sethian

REFERENCES

Anderson, C., and Greengard, C. 1985. On vortex methods. *SIAM J. Numer. Anal.* 22(3):413–440.

Appel, A. W. 1985. An efficient program for many-body simulation. *SIAM J. Sci. Stat. Comp.* 6:85.

Barnes, J. E., and Hut, P. 1986. A hierarchical *O*(*N*log *N*) force-calculation algorithm. *Nature* 324:446–449.

Beale, J. 1986. A convergent 3-d vortex method with grid-free stretching. *Math. Comput.* 46:401–423 and S15–20.

Beale, J., and Majda, A. 1982a. Vortex methods, I: convergence in three dimensions. *Math. Comput.* 39:1–27.

Beale, J., and Majda, A. 1982b. Vortex methods, II: higher order accuracy in two and three dimensions. *Math. Comput.* 39:29–52.

Benz, W., Bowers, R. L., Cameron, A. G. W., and Press, W. H. 1990. Dynamic mass exchange in doubly degenerate binaries I. 0.9 and 1.2 M solar stars. *Ap. J.* 348:647.

Board, J. A., Causey, J. W., Leathrum, J. F., Windemuth, A., and Schulten, K. 1992. Accelerated molecular dynamics simulation with the parallel fast multipole algorithm. *Chem. Phys. Lett.* 198:89.

Choquin, J.-P. 1987. *Simulation numérique d'écoulements tourbillonnaires de fluides incompressibles par des méthodes particulaires.* Ph.D. thesis, Université Paris VI.

Choquin, J.-P., and Cottet, J.-H. 1988. Sur l'analyse d'une classe de méthodes de vortex tridimensionnelles. *C. R. Acad. Sci. Paris* 306(I): 739–742.

Chorin, A. 1980. Vortex models and boundary layer instability. *SIAM J. Sci. Stat. Comput.* 1(1):1–21.

Chorin, A. 1981. Estimates of intermittency, spectra, and blow-up in developed turbulence. *Comm. Pure Appl. Math.* 34:853–866.

Chua, K., Leonard, A., Pépin, F., and Winckelmans, G. 1988. Robust vortex methods for three-dimensional incompressible flows. In *Proc. Symp. recent developments in computational fluid dynamics* 95:33–44, Chicago: ASME.

Cottet, G.-H. 1988. On the convergence of vortex methods in two and three dimensions. *Ann. Inst. Henri Poincaré* 5:227–285.

Ding, H.-Q., Karasawa, N., and Goddard, W. 1992. Atomic level simulations of a million particles: The cell multipole method for coulomb and london interactions. *J. Chem. Phys.* 97:4309–4315.

Dubinski, J., and Carlberg, R. G. 1991. The structure of cold dark matter halos. *Ap. J.* 378:496.

Dyer, C. C., and Ip, P. S. S. 1993. Softening in *N*-body simulations of collisionless systems. *Ap. J.* 409:60–67.

Edelman, A. 1993. Large dense numerical linear algebra in 1993, the parallel computing influence. *Internat. J. Supercomput. Appl.* 7(2):113–128.

Engheta, N., Murphy, W. D., Rokhlin, V., and Vassiliou, M. S. 1992. The fast multipole method (FMM) for electromagnetic scattering problems. *IEEE Trans. Antennas Propagation* 40(6):634–642.

Greengard, L. 1987. *The rapid evaluation of potential fields in particle systems.* Ph.D. thesis, Yale University.

Greengard, L. 1990. Potential flow in channels. *SIAM J. Sci. Stat. Comp.* 11(4):603–620.

Greengard, L., and Rokhlin, V. I. 1989. On the evaluation of electrostatic interactions in molecular modeling. *Chemica Scripta* 29A:139–144.

Hockney, R. W., and Eastwood, J. W. 1981. *Computer simulation using particles.* McGraw-Hill International, New York.

Katz, N., Hernquist, L., and Weinberg, D. H. 1992. Galaxies and gas in a cold dark matter universe. *Ap. J.* 399:L109.

Koumoutsakos, P. 1993. *Direct numerical simulations of unsteady separated flows using vortex methods.* Ph.D. thesis, California Institute of Technology.

Koumoutsakos, P., and Leonard, A. 1992. Direct numerical simulations using vortex methods. In *Proc. NATO advanced research workshop: Vortex flows and related numerical methods,* pp. 15–19, Grenoble, France.

Leonard, A. 1985. Computing three-dimensional incompressible flows with vortex elements. *Ann. Rev. Fluid Mech.* 17:523–559.

Mosher, M. 1985. A method for computing three-dimensional vortex flows. *Z. Flugwiss. Weltraumforsch.* 9(3):125–133.

Novikov, E. 1983. Generalized dynamics of three-dimensional vortical singularities (vortons). *Sov. Phys. JETP* 57(3):566–569.

Pedrizzetti, G. 1992. Insight into singular vortex flows. *Fluid Dynamics Res.* 10:101–115.

Pépin, F. 1990. *Simulation of flow past an impulsively started cylinder using a discrete vortex method.* Ph.D. thesis, California Institute of Technology.

Rehbach, C. 1977. Numerical calculation of three-dimensional unsteady flows with vortex sheets. *Rech. Aérosp.* 5:289–298.

Rokhlin, V. 1985. Rapid solution of integral equations of classical potential theory. *J. Comp. Phys.* 60:187–207.

Saffman, P. 1980. Vortex interactions and coherent structures in turbu-

lence. In *Transition and turbulence,* edited by R. Meyer, pp. 149–166. New York: Academic Press.

Saffman, P. G., and Meiron, D. I. 1986. Difficulties with three-dimensional weak solutions for inviscid incompressible flow. *Phys. Fluids* 29(8): 2373–2375.

Salmon, J. K. 1990. *Parallel hierarchical N-body methods.* Ph.D. thesis, California Institute of Technology.

Salmon, J. K., Quinn, P. J., and Warren, M. S. 1990. Using parallel computers for very large *N*-body simulations: Shell formation using 180k particles. In *Heidelberg conference on dynamics and interactions of galaxies,* edited by R. Wielen, pp. 216–218, New York: Springer-Verlag.

Salmon, J. K., and Warren, M. S. 1994. Skeletons from the treecode closet. *J. Comp. Phys.* 111(1):136–155.

Schmidt, K., and Lee, M. A. 1991. Implementing the fast multipole method in three dimensions. *J. Stat. Phys.* 63(5/6):1223–1235.

Suginohara, T., Suto, Y., Bouchet, F. R., and Hernquist, L. 1991. Cosmological *N*-body simulations with a tree code: Fluctuations in the linear and nonlinear regimes. *Ap. J. Suppl.* 75:631.

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

Warren, M. S., and Salmon, J. K. 1992. Astrophysical *N*-body simulations using hierarchical tree data structures. In *Supercomputing '92,* Los Alamitos: IEEE Comp. Soc.

Warren, M. S., and Salmon, J. K. 1993. A parallel hashed oct-tree *N*-body algorithm. In *Supercomputing '93,* Los Alamitos: IEEE Comp. Soc.

Winckelmans, G. 1989. *Topics in vortex methods for the computation of three- and two-dimensional incompressible unsteady flows.* Ph.D. thesis, California Institute of Technology.

Winckelmans, G., and Leonard, A. 1988. Weak solutions of the three-dimensional vorticity equation with vortex singularities. *Phys. Fluids* 31(7): 1838–1839.

Winckelmans, G., and Leonard, A. 1989. Improved vortex methods for three-dimensional flows. In *SIAM workshop on mathematical aspects of vortex dynamics,* edited by R. Caflisch, pp. 25–35, Leesburg, Virginia: SIAM.

Winckelmans, G., and Leonard, A. 1993. Contributions to vortex particle methods for the computation of three-dimensional incompressible unsteady flows. *J. Comp. Phys.* 109(2): 247–273.

Zhao, F. 1987. An *O(N)* algorithm for three-dimensional *N*-body simulations. M.S. thesis, Massachusetts Institute of Technology.