FITTING THE UNIVERSE ON A SUPERCOMPUTER

Simulations run on the largest available parallel supercomputers are answering the question of how today's rich cosmic structure developed from a smooth, near featureless early universe. Two such simulations illustrate how algorithm and implementation strategies can be optimized for specific cosmological problems.

Stars like our sun group together in the tens of billions to form vast galaxies, each with a complex, but essentially regular, internal structure extending over tens of thousands of light-years. The distances between galaxies are typically a hundred times greater, yet galaxy-distribution maps show—even on such huge scales—dramatic and obvious structures. Many galaxies are in clusters, the most populous containing a thousand or more systems orbiting each other like bees in a swarm. Such clusters form the densest nodes of a more irregular structure, often likened to a web or foam, which defines the boundaries of large "voids" almost empty of galaxies. Only when blurred on scales of hundreds of millions of light-years does the present-day galaxy distribution start to appear uniform.

Nevertheless, we know that the early universe was very nearly uniform. Maps of the cosmic microwave background radiation provide a snapshot of the time when radiation last interacted directly with the universe's matter. This was about 15 billion years ago when the universe was only 300,000 years old. NASA's Cosmic Background Explorer satellite first detected structure—faint ripples—in such maps. In the decade since COBE's launch, however, many other experiments have confirmed these faint ripples, roughly a thousand times weaker than structures of similar extent in present-day galaxy maps. The critical question then is how gravity, due primarily, perhaps, to an entirely new form of matter, caused the present universe's rich and varied texture to grow from a near-uniform initial state. (See the "Dark matter" sidebar for more information about this question and numerical approaches to answering it.).

This article discusses two simulation programs carried out on a Cray T3E in Garching, Germany. The first aims to outline structure in the largest possible volume of the universe. The second treats a single galaxy cluster's evolution with the highest possible resolution. These programs illustrate how algorithm and implementation strategies can be optimized for specific scientific problems. Each has led to the largest simulation of its type yet carried out.

Pushing the envelope

Simulations at the limit of what is currently possible require privileged access to parallel supercomputers and specialized program development to exploit these machines' full capabilities. Cosmological simulations typically generate large output data sets, which are used for a wide range of scientific projects, each requiring different data-reduction and visualization proce-

1521-9615/99/$10.00 © 1999 IEEE

SIMON D.M. WHITE AND VOLKER SPRINGEL
Max Planck Institute for Astrophysics

Dark matter

Gravity, it appears, has played a key role in the universe's transformation. And gravity due primarily to a new kind of matter, different from that which makes up all directly observed objects—from bacteria to PCs to stars. Over the last 25 years, various astronomical observations have led to a consensus that most of the matter in the universe is in some "dark" form, so far observed only indirectly through its gravitational effects—for example, on galaxies in clusters. *Dark matter*, commonly thought to be some new elementary particle, has yet to be detected directly on Earth. It now interacts with other matter almost exclusively through gravity. This idea greatly simplifies cosmological studies. Gravity rules cosmic evolution on large scales, and the dominant source of gravity is dark matter. Thus we can follow evolution in the dark part of the universe without worrying about the other physics (shocks, radiative cooling and heating, star formation, and so on) that affect ordinary matter. The rub, of course, comes when comparing the results with real data. What we see is the ordinary matter, not the dark matter.

Dark matter motions

The physical equations governing the dark-matter particles' motion are extremely simple:

$$m_i \frac{d^2\mathbf{x}_i}{dt^2} = -\sum_{i \neq j} \frac{G m_i m_j \mathbf{r}_{ij}}{r_{ij}^3} \qquad \text{for } i = 1, \ldots, N.$$

Because all motions remain much slower than light, we can neglect the complications of Einstein's general relativity in favor of Newton's simpler theory of gravity. To solve these equations, we must, of course, specify both the particles' initial positions and velocities and the conditions at the boundary of the region being studied. Current theories for the origin of structure provide the initial conditions. These theories suggest that deviations from exact uniformity result from zero-point fluctuations in the quantum field that drove a rapid "inflationary" expansion about 10^{-30} seconds after the big bang. Boundaries are usually treated either by assuming external regions to be periodic replicas of the prime region (normally taken to be a cube) or by following some vicinity of the prime region at reduced resolution and then neglecting the effects of even more distant regions.

Practical problems

The real limitation in using these equations to simulate structure growth results from the values of N required. This is because we must integrate N vector equations, each of which has $N - 1$ separate force terms on its right-hand side. The total mass of dark matter in the observable universe is roughly 10^{55}g, and the mass of an individual dark-matter particle might be around 10^{-21}g, giving $N \sim 10^{76}$. Of course, a simulation clearly does not have to follow all dark-matter particles, only a representative Monte Carlo sampling of them. If, however, we imagine representing the mass of a typical galaxy like our own (~10^{44}g) with just 10 simulation particles, we would still require $N \sim 10^{12}$. A full simulation using the equations of motion in the form above would then need on the order of 10^{28} flops, and storing the data at just the initial (or final) time would require tens of Tbytes.

Clearly, efficient strategies are needed to reduce the operation count associated with calculating the forces and to reduce N to values where data storage is feasible. The first problem is essentially algorithmic: what is the minimum operation count needed to achieve some desired accuracy in the force calculation? The second depends on the specific astrophysical question under consideration: what is the smallest region that can be considered a fair sample of the universe for the problem, and how massive can individual simulation particles be before their discrete nature begins to compromise the structure of the objects of interest? Over the years, algorithm design aimed specifically at the gravitational N-body problem has led to several near-optimal solutions for the first problem. The second problem continues to push toward code implementation on the largest available computers.

dures. These factors have pushed numerical cosmologists into relatively large collaborations. In the United States, two such Grand Challenge consortia are active, one based primarily on the East Coast and one on the West. In Europe, the Virgo Consortium, a grouping of scientists from Britain, Germany, Canada, and the US, uses machines at the Edinburgh Parallel Computing Centre and at the Computing Centre of the Max-Planck Society at Garching, Germany.

The next generation of observational surveys will map the 3D distribution of galaxies over a significant fraction of the sky out to distances of about two billion light-years. The most ambitious of these projects is the Sloan Digital Sky Survey, discussed in Alex Szalay's article in this issue. To obtain realistic theoretical predictions for what such projects might see, you need to simulate structure formation in a region substantially larger than the one surveyed. Only then can you construct a number of independent artificial galaxy surveys and, by comparing them, form an opinion about the likelihood that the real survey will find structures of any given type—for example, very large walls or voids, or a population of very massive galaxy clusters. This need

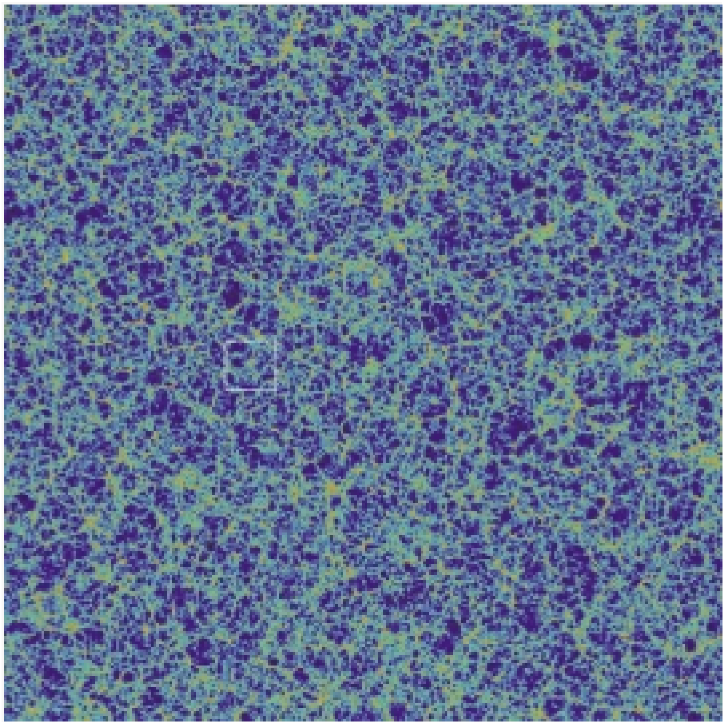
Figure 1. The present-day mass distribution in a thin slice through one of the Hubble Volume simulations. The box shown is 13 billion light years on a side. The brightest structures in this picture correspond to rich clusters of galaxies and to the filamentary web connecting them. Thousands of such clusters exist in the Hubble simulation, each being a system containing up to several thousand galaxies. Due to the simulation's huge volume, the general texture looks rather regular. Only when blurred on such large scales does the universe appear homogenous. The white square gives the relative size of Figure 3's intermediate-scale simulation.

prompted the Virgo Consortium, following a suggestion from Gus Evrard of the University of Michigan, to modify their simulation codes to follow structure in the largest possible regions given the available computer resources. These turned out to be cubes 13 billion light-years on a side, regions so large that they correspond to a significant fraction of the entire observable universe, the *Hubble Volume*.

Hubble Volume simulation

The basic algorithms underlying the Hubble Volume code were taken from public-domain software written originally for serial machines by Hugh Couchman of the University of Waterloo. This Adaptive Particle-Particle/Particle-Mesh (AP^{3}M) code speeds up the force calculation by splitting scales. The forces due to the large-scale distribution of matter are obtained by scattering the particles onto a regular cubic mesh (1,024^3 was used for the Hubble Volume simulations) to create a smoothed density field. Fast Fourier transforms (FFTs) are then used to solve Poisson's equation for the mesh's gravitational potential, and the result is differenced and interpolated to each particle's position to give the force due to the smoothed density field. A second step adds the difference between this smoothed force and the more accurate inverse-square force that is desired. The algorithm's efficiency lies in the fact that this difference depends only on the matter distribution local to each particle and so can be evaluated by summing contributions from a relatively small number of neighboring particles. In Couchman's serial AP^{3}M, the correction is evaluated by a direct sum over neighbors in regions where the number of particles per mesh cell is relatively small and by adding local refinement meshes in regions where this number is large. In the Hubble Volume simulations, the particle distribution is uniform enough so that refinement meshes are not needed, sparing the associated memory for other purposes.

For this Particle-Particle/Particle-Mesh algorithm (P^{3}M),[1] we can estimate the operation count for a cosmological simulation with N particles and an M^3 mesh as follows. Creating the density mesh, differencing and interpolating the potential to obtain smoothed forces, and advancing the particles are all associated with an operation count that scales as N. The FFTs associated with computing the grid potential have an operation count proportional to $M^3 \ln M^3$. The calculation of the short-range force corrections scales with $N \times N_{\text{neigh}} = N^2/M^3$, where N_{neigh}, the typical number of neighbors, is inversely proportional to M^3 because a finer mesh requires fewer neighbors. In practice, minimizing the overall operation count for a given N means choosing M so that the force calculation's particle-mesh (PM) and particle-particle (PP) parts take similar amounts of time. For the Hubble Volume simulations where $N = 10^9$, it turned out that $M = 1{,}024$ gave good performance. For these calculations, where the large-scale particle distribution is nearly uniform and the most massive individual structures contain a few thousand particles, it suffices to advance all the particles with the same time step using a straightforward leapfrog integrator:

$$\mathbf{v}_{i+\frac{1}{2}} = \mathbf{v}_{i-\frac{1}{2}} + \frac{\mathbf{F}(\mathbf{x}_i)}{m_i} \times \Delta t; \;\; \mathbf{x}_{i+1} = \mathbf{x}_i + \mathbf{v}_{i+\frac{1}{2}} \times \Delta t.$$

This requires only one force calculation per time step and minimizes the storage space needed. Somewhat less than 600 time steps were needed to bring the Hubble Volume simulations from the epoch of recombination up to the present day.

Parallelizing the universe

The adaptation of the Virgo simulation codes for the Hubble Volume project was carried out by Tom MacFarland, then on staff at the Garching Computer Center, in close consultation with Hugh Couchman, Jakob Pichlmeier (Cray/SGI Germany), and Frazer Pearce (University of Durham, UK). Hugh wrote the original serial code, Frazer made an earlier parallelization of it, and Jakob was familiar with many of the intricacies of programming for the Cray T3E. Two factors were critical in creating a code capable of carrying out the largest feasible calculation. The first was to find a mapping of the particle data onto the machine that optimized the distribution of work across processors. In practice, different schemes were employed for the PM and PP parts of the force calculation, with a data redistribution step between them. The second was to minimize the storage required both by the basic algorithm (the leapfrog stores only the particles' current positions and velocities, a set of link lists used to locate neighbors, a single scalar function on the mesh, and a smaller number of auxiliary quantities) and by the duplicate information needed to communicate particle positions between processors when locating neighbors. The solutions that MacFarland and his colleagues finally adopted made it possible to follow 10^9 particles with a $1{,}024^3$ mesh on 512 processors of the Garching T3E.[2] The full 128 Mbytes of memory on each processor was used with very little redundancy.

Interesting complications

Outputting and storing the data from the Hubble Volume simulations also led to some interesting problems. A single snapshot of the positions and velocities of all the particles requires 26 Gbytes, and seven such snapshots were kept for each simulation to preserve a record of its evolution. Figure 1 shows a thin slice through such a snapshot at the present time. The snapshots cannot, however, be compared directly with deep observational surveys because the finite speed of light is important over such large distances. When observational astronomers map structures that are 10 billion light-years away, they map these structures not as they are today, but as they were 10 billion years ago when the light left. Because our universe is only about 15 billion years old, observed distant structures are younger and less developed than those which are seen nearby. When suggesting the Hubble Volume project, Gus Evrard stressed the importance of saving simulation data along the past light cones of a number of "observers" so that this aspect of the observations could be mimicked properly, and he took responsibility for coding the relevant output routines. As a result, in the simulations, as in the real universe, it is possible to see evolution directly by looking at how structure changes with distance in a single survey (see Figure 2 for a picture of the Hubble light cone). With this modification, the total amount of raw data generated by a single Hubble volume simulation grew to 0.7 Tbytes. Because these data are produced in 70 to 100 hours of CPU time, the dataflow rate was very large and seriously stressed both Joerg Colberg, the scientist managing the runs, and the Garching Computer Center's archiving system.

Tracing the web

Two Hubble Volume simulations were completed during the project's

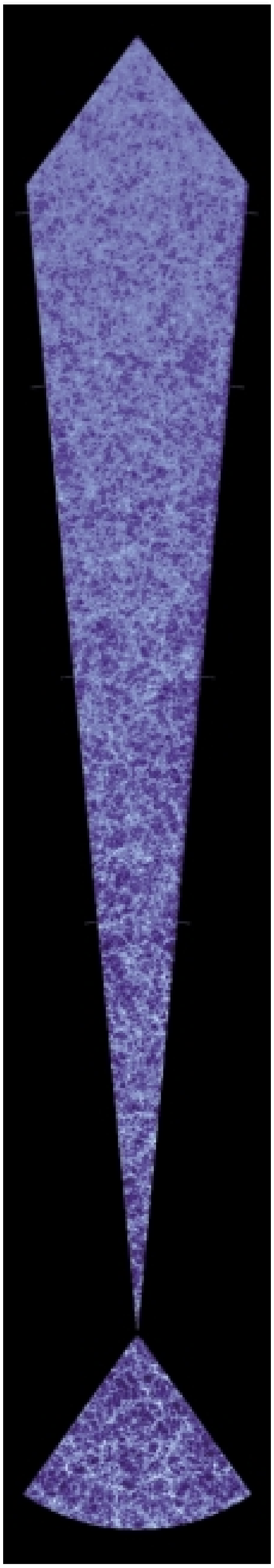

Figure 2. The Hubble Volume light cone. While the Hubble simulation was run, particle information was continuously accumulated along the past light cone of a fiducial observer. As a result, the simulated universe's evolution directly manifests itself in the "observed" structure's changes as a function of distance. Just like in the real universe, looking to larger distances means traveling further backward in time. The light cone thus allows a direct comparison between the simulation and deep observations of the universe. The observer at the present time is located at the cone's apex.

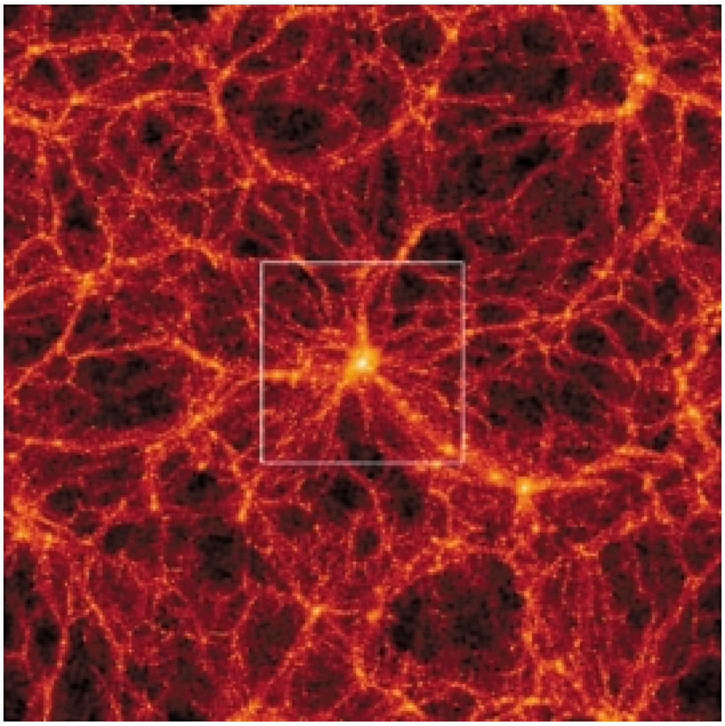
Figure 3. A thin slice through a simulation with only 16.7 million particles, but with much smaller volume and 200 times better mass resolution than the Hubble Volume simulations. For this picture, the periodic simulation box has been centered on the most massive cluster of galaxies that formed in this simulation. The cluster is surrounded by a web of filaments from which most of its mass was accreted during its formation. We selected the region in the white square as a target for resimulation at a much higher resolution (see

first phase, each producing about 0.4 Tbytes of reduced and archived data. The simulations differ in the total amount of dark matter the universe is assumed to contain. In one, the average dark-matter density is assumed sufficient for its gravity eventually to reverse the cosmic expansion. In the other, the amount of dark matter is three times smaller, and the expansion is actually accelerating, at present, as a result of "pressure" from the inclusion of a cosmological constant in Einstein's theory of gravity. Recent observations of distant supernovae suggest that the real universe might indeed be accelerating just as this theoretical model suggests. The first comparisons of these simulations with each other and with real data are now complete. The remarkable quantitative agreement with observation already found on smaller scales seems to continue over the much larger distances that these new simulations can probe. In addition, the observed evolution of massive structures such as galaxy clusters seems in better accord with the simulation with a cosmological constant, providing independent confirmation of the supernova results. The archived data from these simulations will provide a wealth of detailed material for comparison with the Sloan Survey and other surveys like it. Mining the archive requires top-end workstations and specially tailored visualization and data-processing techniques because of the individual data sets' large size.

The Hubble Volume simulations' texture appears quite homogeneous because of the very large region followed. A zoom into a smaller region shows, however, that the theoretical model predicts a very rich small-scale structure. Figure 3 shows a thin slice through a simulated cubic region 21.3 times smaller than the corresponding Hubble Volume simulation. This small simulation followed $N = 256^3$ particles, so the mass of each was 200 times smaller than in the Hubble Volume model. Rich clusters of galaxies are then represented by clumps of several tens of thousands of particles rather than by just a few hundred, and far fewer massive condensed objects are visible. The smallest of these contain a few tens of particles and correspond to the dark-matter distribution around an individual isolated galaxy like our own. They are clearly linked to each other and to the clusters in a complex web-like pattern. To follow the internal structure and the formation path of these "galaxy halos," and hence to understand how their evolution might shape the formation of the galaxies in them, the simulations' mass and spatial resolution must be improved by two orders of magnitude. As we now discuss, making such calculations feasible requires different numerical techniques.

Galaxy cluster simulation

Clusters of galaxies are the most massive quasi-equilibrium objects known. The biggest systems have masses of 10^{48}g or more and can contain several thousand galaxies. According to the standard theoretical paradigm, each of these galaxies formed at an early epoch inside its own isolated dark halo. The cluster itself grew later as galaxies and halos flowed along the filaments and sheets, evident in Figure 3 to merge together at the intersections. A proper simulation of this process must correctly represent both the halos' internal structure and the large-scale pattern of sheets and filaments. The latter span scales of many tens of millions of light-years, while galaxies themselves are a few thousand light-years across. Furthermore, if the mass in the visible part of a small galaxy is to be represented by, say, a hundred par-

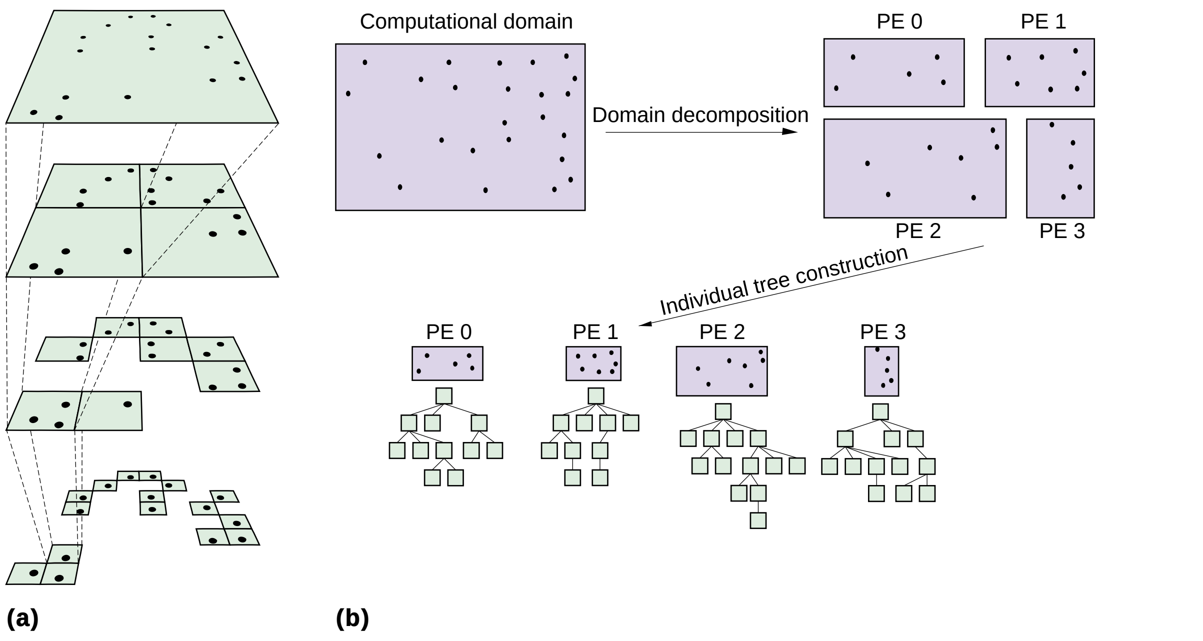

Figure 4. (a) A schematic illustration of the Barnes & Hut tree (for clarity just drawn in two dimensions). The root node is a box encompassing all the particles. Each tree node is then recursively subdivided into eight nodes, each with half the side-length of its parent, until a node contains a single particle. (b) The orthogonal recursive domain decomposition employed in Gadget to spatially partition the computational volume onto the individual processors. Each processor then constructs a local Barnes & Hut tree, which is used in the force computation to provide partial forces at arbitrary locations in space.

ticles, you need about 10^7 particles to represent the entire mass in a cluster's quasi-equilibrium region. The Hubble Volume code is very poorly suited to such a problem because it is designed for situations where all particles have similar environments, require similar effort to evaluate their acceleration, and can be advanced with similar time steps. For a cluster simulation, the P^3M technique becomes very slow because of the very large number of neighbors that must be included for particles in dense regions. In addition, the wide range of densities in the cluster is associated with a wide range of orbital time scales for individual particles. In this situation, an integration scheme that implements individual particle time steps that adapt to local conditions can greatly reduce the operation count.

A gadget for simulating clusters

Various algorithms have been developed for the cluster problem, mostly based on the notion of *zooming in* onto the region of interest. The density field is sampled with high resolution only where the material will become part of the final cluster and its immediate surroundings. The cosmological environment on larger scales is represented by heavier particles in such a way that the resolution degrades with distance from the cluster. This resolution hierarchy is supplemented by a hierarchical algorithm for computing gravitational forces. In *tree codes*, this is based on the observation that the gravitational field of a distant group of particles can be well approximated by the first few terms in a multipole expansion. With this idea in mind, you can construct a hierarchical grouping of the particles in the form of a tree. The tree's root is a group encompassing all the particles. Branches lead from this node to other nodes, each corresponding to a smaller, spatially localized group. This branching continues to smaller and smaller groups, ending finally in leaves corresponding to single particles. The force computation for a particular particle proceeds by walking the tree, starting at its root. Once a sufficiently small and distant node is reached, a multipole representation evaluates its force contribution to the particle and the tree walk is terminated along this branch. Otherwise, the walk proceeds to all of the node's subnodes. A complete force evaluation for all particles requires on the order of $N\log(N)$ multipole terms—a substantial saving compared to the N^2 scaling of the naive direct-summation approach.

There are several ways to construct a suitable tree in practice. In our code Gadget (*ga*laxies with *d*ark matter and *g*as int*e*rac*t*), we use a popular geometrical construction introduced by Barnes & Hut.[3] As sketched in Figure 4, the root node is taken to be a cube encompassing all particles. This cube is recursively subdivided into eight cubes, each with half the side-length of its parent, resulting in an oct-tree structure.

Tree algorithms are not intrinsically restricted in their spatial dynamic range, nor in the particle system's geometry. Also, in strong contrast to the P^3M method, a force computation's cost depends only weakly on the strength of clustering. Furthermore, in systems with vastly different dynamical timescales, the tree algorithm can be tied into an integration scheme with individual particle time steps such that the force computation for a subset of M particles takes on the order of $M\log(N)$ operations. Here, the salient point is that there is no term in the computational cost that scales proportional to N—it would eventually dominate the runtime for $M \ll N$, which is expected if the dynamic range in time is large and individual particle time steps are used.

You can achieve this favorable scaling of the force algorithm by realizing that the tree does not have to be reconstructed from scratch for every small time step. Because the multipole moments of larger groups tend to evolve more slowly than those of smaller ones, it often suffices to make small predictions of these moments, or to update only certain parts of the tree. This can be done while the tree is walked, and only involves the nodes that are actually needed in the force computation. Finally, the code employed to select and advance the small particle group of size M can also be freed of any overhead proportional to N. To this end, Gadget's integration scheme organizes the particles in a priority queue, where particles can be extracted and reinserted with an operation count on the order of $\log(N)$. At each time step, a group of size M is selected from the head of the queue. These are the particles with highest priority—that is, the ones that need to be advanced next.

Combined with such an individual time-step integration scheme, tree algorithms provide an efficient tool to bridge a large dynamic range in the gravitational N-body problem. Alternative Poisson solvers for such highly clustered particle distributions include the publicly available adaptive P^3M code discussed earlier as well as adaptive mesh refinement codes, which are entirely mesh-based (see Greg Bryan's article in this issue). AP^3M places refinement meshes over regions where the PP workload is particularly high, allowing FFT techniques to calculate the longer-range part of the PP correction; direct neighbor corrections are then needed over a much smaller region around each particle. AMR refines the mesh in high-density regions in a similar way to the Barnes-Hut oct-tree discussed previously. The Poisson equation is solved directly for the density defined on the refined mesh to give a potential on this same mesh. Both techniques are promising although neither has yet been used for a cluster simulation as large as the one we present below. A disadvantage of tree algorithms is that they require substantially more memory than plain P^3M codes. As a result, on most systems, memory limitations define a simulation's maximum feasible size.

Problems with parallel trees

On massively parallel supercomputers, you face additional challenges: memory that is physically distributed onto the computational nodes, and the fact that no single node will usually have enough space to hold a copy of the full tree. John Dubinski (Canadian Institute for Theoretical Astrophysics) described a successful parallelization strategy for a tree code under these circumstances.[4] Our code Gadget adopts his strategy for partitioning the problem onto the computational nodes (see Figure 4), but computes the force differently to allow the efficient use of individual time steps.

In orthogonal recursive bisection, you first decompose the computational volume into disjoint domains that are each mapped onto a single processor; this processor stores all the particles inside the domain. In practice, the volume is cut parallel to each coordinate plane in such a way that the number of particles on the two sides of the cut (or, as we explain later, the work required to advance them) is equal. This initial decomposition can be viewed as the first layer in a global tree. As illustrated in Figure 4, each domain is then itself the root of a local tree, which each processor can construct independently of all the others.

Note that it is really "only" the force computation that requires information from more than one domain. All other parts of the simulation algorithms are intrinsically parallel. The information needed from each domain is just the force exerted by it on some particle position. To compute the force on a given particle, you could therefore communicate its coordinates to all processors, which then walk their local trees to obtain partial forces exerted by the mass distribution inside their

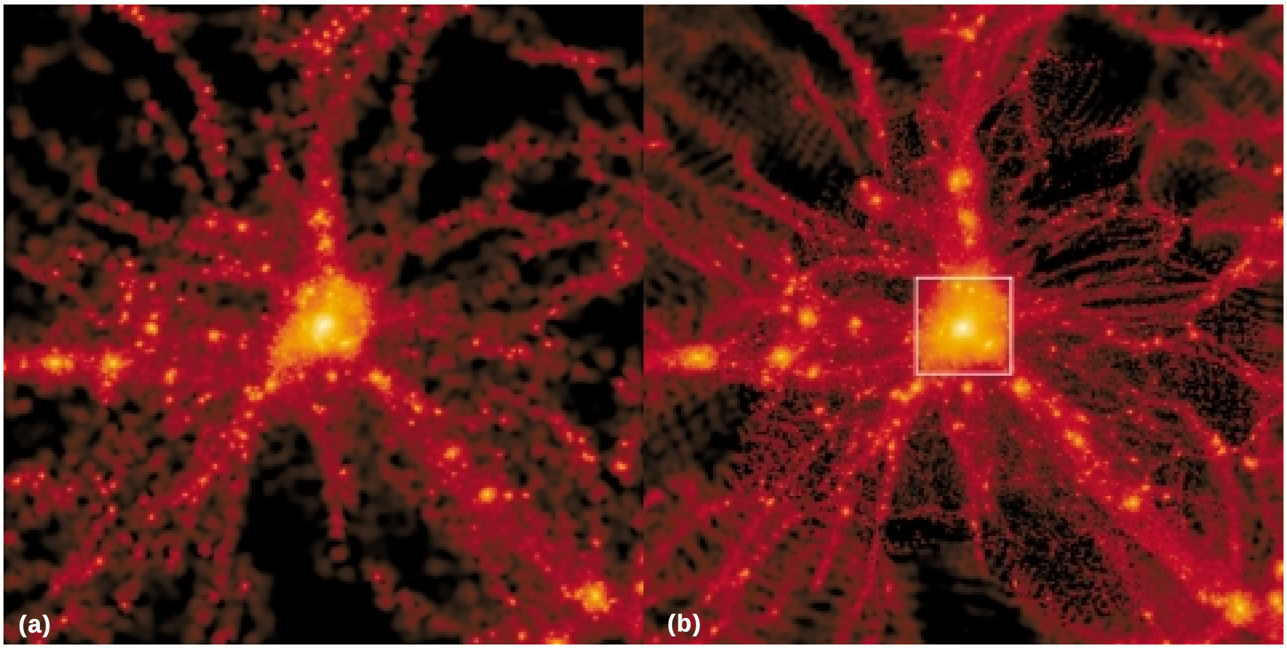

Figure 5. (a) The target cluster and its environment in the intermediate-scale simulation displayed in Figure 3. For comparison, we also show (b) our resimulation of this object. Here, the mass resolution is 300 times higher in the center of the simulation volume, where the cluster itself forms. We selected the region inside the white square for further enlargement in Figure 6.

domains, and finally these forces are summed up and communicated back to the processor that sent out the particle position. However, if this were done just for a single particle, the work done by the processors would be badly imbalanced. A processor with a distant domain might be able to stop its tree walk right at the root node, but closer domains would have to traverse the tree to an ever-increasing depth. Consequently, many processors would have to wait idly until the one with most of the work finishes, losing most of the parallel computer's computational power.

To remedy this situation, Gadget processes the force computation by using lists of particle coordinates containing a representative mix of all the particles in the simulation. At a given time step, each processor contributes some particles to this list, which is then established in a collective communication process on all computational nodes. They then walk their local trees independently, and finally participate in a collective summation-and-communication phase that delivers the total force on each particle to the processor that originally contributed the corresponding coordinate to the list. Because the tree walk is by far the most time-consuming part of the simulation, the issue of workload balance is most crucial here. By measuring the work incurred by individual nodes and particles in the local trees, the code assigns a cost factor to each particle. The domain decomposition then repeats at regular intervals using this information to balance the total "cost" between domains. This provides an excellent overall workload balance, which adjusts dynamically to the system's evolution.

A very big cluster

The largest cluster simulation so far carried out with this (or any other) *N*-body code was a resimulation at much higher resolution of a massive cluster from the intermediate-scale simulation illustrated in Figure 3. The particular cluster chosen is seen at the figure's center. To create initial conditions for the new simulation, Bepi Tormen (University of Padova) and we identified all particles in the original cluster and its surroundings and traced out the region of the initial conditions from which they came. For the new initial conditions, the matter distribution in this region was represented with much higher resolution than before (and by a much larger number of particles, about 6.6×10^7), while the resolution in the rest of the original volume was progressively degraded as the distance from the cluster increased. Starting from these new initial conditions, we could follow the cluster's for-

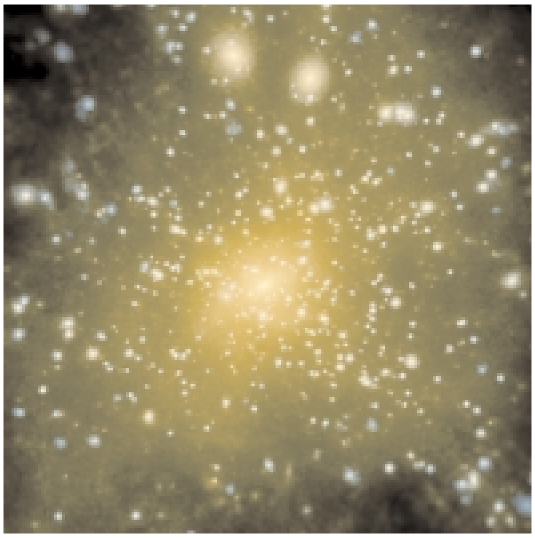

Figure 6. The final cluster of galaxies at the present day. The displayed region is a cube of 25 million light-years on a side around the cluster center. While some substructure can be seen even in Figure 5, the present visualization has been specifically designed to highlight it. As a result, the galaxies orbiting in the cluster can be seen beautifully.

mation with greatly improved resolution while correctly representing the tidal fields due to neighboring structures (see Figure 5). More than 20 million particles end up in the final cluster's quasi-equilibrium region.

In Figure 6, we show a visualization of this object designed to highlight its very rich substructure, which is made up of many distinct gravitationally bound clumps of dark matter orbiting independently in a general diffuse sea of dark matter. This diffuse component actually contains most of the cluster mass. The clumps are remnants of individual dark-matter halos that formed at early times and later fell into the forming cluster. The cluster's gravitational tides tear most of the mass off these halos, but their cores survive and plausibly mark the locations of the galaxies that should have formed at their centers before they fell into the cluster.

To our knowledge, this cluster simulation achieves the highest dynamic range in mass and length ever reported for a cosmological *N*-body simulation. The smallest (3×10^3 l-yr) and largest (6×10^8 l-yr) resolved length scales differ by a factor of 2×10^5. An individual particle's mass in the high-resolution region is about 10^{-10} of the total mass of the region in which structure formation is followed. Individual particle time steps in the densest regions are as small as 5×10^{-6} of the total time span simulated. The calculation took more than 300 hours to run on 512 processors of the Garching T3E and used essentially all of the 65 Gbytes of memory available.

The analysis of this extremely well-resolved cluster has just begun. Ben Moore of Durham University in collaboration with colleagues in Durham and Seattle has extensively studied simulations with about an order-of-magnitude poorer resolution. Their numerical data were already an enormous advance on anything previously available and showed much of the rich structure visible in our figures. The present simulation probably reaches the limit of what is sensible for studies of an individual cluster because its resolution limits, both in mass and in length, are significantly smaller than the scales on which nongravitational effects acting on ordinary matter are known to have a major impact on galaxy structure. Our cluster simulation, like the Hubble Volume simulations, also nears the limit where extensive postprocessing ceases to be viable. For example, storing just a single snapshot of the cluster requires about 1.7 Gbytes, and a detailed study of its formation history requires a comparative analysis of about 50 such snapshots. The simulation's data archive is thus about 85 Gbytes.

From our work, it seems that the frontier in cosmological *N*-body studies is currently defined more by the ability to handle very large numerical data sets than by the need to carry out very large numbers of floating-point operations. Both the Hubble Volume simulations and our cluster simulation filled all the memory on the largest machine available to us, yet could be completed using relatively small amounts of CPU—at least when judged by the standards of Grand Challenge problems! In both cases, difficulties with data manipulation, visualization, and reduction primarily limit scientific exploitation of the data archives. Several years of fruitful work remain to be done on a variety of projects with both archives, and both will be made available to interested cosmologists with sufficient computational resources to be able to use them. As with large observational surveys, the scientific harvest from such data sets is limited primarily by the imagination and ingenuity of the astronomers who work on them.

References

1. R.W. Hockney and J.W. Eastwood, *Computer Simulation Using Particles*, McGraw-Hill, New York, 1981.
2. T. MacFarland et al., "A New Parallel Code for Very Large-Scale Cosmological Simulations," *New Astronomy*, Vol. 3, No. 8, 1998, pp. 687–705.
3. J. Barnes and P. Hut, "A Hierarchical $O(N\log N)$ Force Calculation Algorithm," *Nature*, Vol. 324, Dec. 1986, pp. 446–449.
4. J. Dubinski, "A Parallel Tree Code," *New Astronomy*, Vol. 1, No. 2, 1996, pp. 133–147.

Simon D.M. White is the managing director of the Max Planck Institute for Astrophysics in Garching near Munich. He received his PhD from Cambridge University and has held research and teaching positions in Berkeley, Paris, Tucson, Cambridge, and Munich. His research interests include extragalactic astronomy and cosmology—particularly, the simulation and formation of galaxies, galaxy clusters, and larger structures. His PhD thesis contained the first published simulation, using just 700 particles, of the hierarchical buildup of a galaxy cluster. Contact him at the Max Planck Institut für Astrophysik, Karl-Schwarzschild-Str. 1, D-85740 Garching bei München, Germany; swhite@mpa-garching.mpg.de; http://www.mpa-garching.mpg.de/~swhite/.

Volker Springel is a PhD student at the Max Planck Institute for Astrophysics, Germany. His research interests include extragalactic astronomy and cosmology—in particular, the large-scale structure of the universe, the formation and evolution of galaxies, and the numerical simulation of these processes, both with collisionless and hydrodynamical techniques. He received his Diploma in physics from the University of Tübingen, Germany. After completing his PhD this summer, he will start his postdoctoral research at the Harvard-Smithsonian Center for Astrophysics. Contact him at the Max Planck Institut für Astrophysik, Karl-Schwarzschild-Str. 1, D-85740 Garching bei München, Germany; volker@mpa-garching.mpg.de; http://www.mpa-garching.mpg.de/~volker/.