Compression of Particle Data from Hierarchical Approximate Methods

DOW-YUNG YANG
Information Systems Laboratories, Inc.
ANANTH GRAMA
Purdue University
VIVEK SARIN
Texas A&M University
and
NAREN RAMAKRISHNAN
Virginia Tech.

This article presents an analytical and computational framework for the compression of particle data resulting from hierarchical approximate treecodes such as the *Barnes–Hut* and *Fast Multipole Methods*. Due to approximations introduced by hierarchical methods, various parameters (such as position, velocity, acceleration, potential) associated with a particle can be bounded by distortion radii. Using this distortion radii, we develop storage schemes that guarantee error bounds while maximizing compression. Our schemes make extensive use of spatial and temporal coherence of particle behavior and yield compression ratios higher than 12:1 over raw data, and 6:1 over gzipped (LZ) raw data for selected simulation instances. We demonstrate that for uniform distributions with 2M particles, storage requirements can be reduced from 24 MB to about 1.8 MB (about 7 bits per particle per timestep) for storing particle positions. This is significant because it enables faster storage/retrieval, better temporal resolution, and improved analysis. Our results are shown to scale from small systems (2K particles) to much larger systems (over 2M particles). The associated

A. Grama's work was supported in part by National Science Foundation (NSF) grants EIA-9806741, ACI-9875899, and ACI-9872101.
V. Sarin's work was supported in part by NSF grant CCR-9972533.
N. Ramakrishnan's work was supported in part by NSF grants EIA-9974956 and EIA-9984317.
Computing equipment used for this work was supported by NSF MRI grant EIA-9871053 and by the Intel Corporation.
Authors' addresses: D.-Y. Yang, Information Systems Laboratories, Inc., 11140 Rockville Pike, Suite 500, Rockville, MD 20852, email: dyang@islinc.com; A. Grama, 1398, Computer Sciences Building, Purdue University, West Lafayette, IN 47907, email: ayg@cs.purdue.edu, web: http://www.cs.purdue.edu/people/ayg; V. Sarin, Department of Computer Science, Texas A&M University, College Station, TX 77843-3112, email: sarin@cs.tamu.edu, web: http://www.cs.tamu.edu/faculty/sarin; N. Ramakrishnan, Department of Computer Science, Virginia Tech., Blacksburg, VA 24061, email: naren@cs.vt.edu, web: http://www.cs.vt.edu/~ramakris.

algorithm is asymptotically optimal in computation time ($O(n)$) with a small constant. Our implementations are demonstrated to run extremely fast—much faster than the time it takes to compute a single time-step advance. In addition, our compression framework relies on a natural hierarchical representation upon which other analysis tasks such as segmented and window retrieval can be built.

Categories and Subject Descriptors: E.2 [**Data**]: Data Storage Representations; E.4 [**Data**]: Coding and Information Theory—*data compaction and compression*; G.1.0 [**Numerical Analysis**]: General—*error analysis*; G.1.10 [**Numerical Analysis**]: Applications; H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing—*abstracting methods*; J.2 [**Computer Applications**]: Physical Sciences and Engineering—*physics*

General Terms: Algorithms, Design

Additional Key Words and Phrases: Astrophysics, Barnes–Hut, data compression and analysis, Fast Multipole Method, materials simulation, molecular dynamics, particle dynamics

1. INTRODUCTION AND MOTIVATION

Particle methods find application in a variety of domains ranging from molecular dynamics to astrophysics. Starting from an initial state, the system state is advanced by computing forces (such as Coulombic and Lennard-Jones) at each timestep and advancing the particles using a leapfrog scheme. A simple all-to-all force computation in an n particle system results in a complexity of $O(n^2)$ because of long-range Coulombic (or gravitational) forces. A number of approximation techniques have been explored to reduce this complexity. Of these, the prominent ones are lattice-based methods and hierarchical treecodes such as Barnes–Hut [Barnes and Hut 1986] and Fast Multipole Method (FMM) [Greengard and Rokhlin 1987]. This article focuses on compression of particle data resulting from simulations based on hierarchical treecodes.

In typical particle dynamics simulations, each particle has, associated with it, a number of parameters such as position, velocity, acceleration, and electrostatic or gravitational potential. These parameters are computed at each timestep, and ideally, are stored for analysis. Analysis tasks range from simple tasks such as segmented retrieval and window retrieval, to more complex tasks such as identifying the onset of cracks and fissures in material simulations, to even more complex tasks such as deriving empirical relationships from data (e.g., relating thermal conductivity and density, etc.). One of the ultimate goals of these simulations is to develop techniques for identifying simulation parameters that lead to desirable material properties. The offline nature of most of these analysis tasks requires effective techniques for storage, handling, and retrieval of simulation data.

It is easy to see that particle simulations can generate extremely large amounts of data. Simply storing position of each particle at every timestep requires 12 bytes per particle per timestep (4 bytes/float). For a modest 100K particle system over a million timesteps, this data corresponds to roughly a terabyte of storage. Merely storing, retrieving, and transferring this data can pose formidable challenges, let alone running meaningful analysis tasks on them. Due to this high storage requirement, data is typically stored once every k

timesteps, where k is selected based on available storage, underlying phenomena being analyzed, and timestep size. This places severe restrictions on the post-processing operations that can be performed on the data since phenomena at lower time-scales are completely lost in temporal subsampling. There is a pressing need for storage and representation techniques that reduce the amount of data storage and bandwidth while supporting analysis tasks. Although considerable work has gone into compression techniques for both topology and geometry data associated with meshes [Taubin and Rossignac 1998; Bajaj et al. 1999; Levoy 1995; Deering 1995], the unique characteristics of this problem combined with theoretical error bounds of hierarchical methods provide us with unique opportunities for effective compression. It is important to note that the most effective compression technique is the particle dynamics simulator itself. The initial state of the problem corresponds to the compressed data and the decompressor is the simulation program. However, the computation associated with the decompressor makes this view impractical.

Hierarchical multipole methods (both FMM and Barnes–Hut variants) use a truncated series approximation of charges within a localized region to estimate impact on well-separated sets of particles. The method of Barnes and Hut relies only on particle—cluster interactions to achieve an $O(n \log n)$ computational bound for uniform particle distributions. The Fast Multipole Method uses both particle—cluster and cluster—cluster interactions to achieve an $O(n)$ bound for uniform distributions. We refer the readers to the work of Barnes and Hut [1986] and Greengard and Rokhlin [1987] for detailed description of these methods. For nonuniform distributions, similar bounds can be obtained by using box-collapsing techniques of Callahan and Kosaraju [1992] or the chaining techniques of Aluru and Gustafson [1996].

As we shall further elaborate in Section 2, one of the major advantages of these methods is their ability to bound the error associated with approximations introduced by them. Multipole methods are used to compute the forces and/or potentials at each timestep. These forces are used to advance particle positions using methods such as the leapfrog scheme. The error associated with multipole methods induce an error in other particle parameters such as position, velocity, energy, etc. This induced error, or distortion sphere forms the basis for our compression schemes. Our compression schemes position the particle within specified distortion radius to maximize compression. Specifically, they place the particle appropriately within a sphere of radius $(1 + \mu)d$, centered about the accurate particle parameter. Here d is the distortion radius induced by the hierarchical treecode and μ is a user defined parameter (usually selected to be less than 0.01). The proposed methods also makes extensive use of spatial and temporal coherence of particle behavior to improve compression. Using these, we develop a family of schemes that reduce the storage-per-particle to under 8 bits/particle per timestep in the best case. This corresponds to a compression ratio of over 12:1 over raw data and over 6:1 over gzipped raw data (LZ) [Ziv and Lempel 1977, 1978] for selected simulation instances. In addition to excellent compression ratios, we demonstrate the following desirable properties of our compression scheme:

—Bounded error rates.
—Low compression and decompression times—both $O(n)$ with small constants.
—Fast querying of intermediate timesteps and retrieval of specified subdomains over specified timesteps.
—In-built framework for analysis of spatial and temporal artifacts in data.

The rest of this paper is organized as follows: In Section 2, we outline the theoretical basis for various error bounds; in Section 3, we describe a family of compression schemes based on our framework; Section 4 presents compression ratios and timings for these schemes, and the impact of proposed schemes is discussed in Section 5.

2. THEORETICAL UNDERPINNINGS

Consider a system of n charged particles with charges $q_0, q_1, \ldots, q_n$. Significant developments in the use of hierarchical multipole-based approximations have enabled simulation of such systems for large values of n. These approximations result in bounded errors in potentials and forces, and consequently derived parameters such as particle velocities and positions that are used to advance system state. This bounded error, also known as a distortion radius, can be used to effectively reduce the storage associated with the state of the system at intermediate timesteps. In this section, we summarize a sequence of analytical results that provide bounds on various particle parameters that are used for compressing particle data.

Given a function evaluation f and an approximation of the evaluation f_A, conventional metrics of relative error compute ϵ as

$$\epsilon = \frac{|f - f_A|}{|f|}.$$

Here, $f \in \Re^m$ is an m-tuple of reals. The associated distortion radius D is given by:

$$D = |f - f_A| = \epsilon |f|.$$

Given f and ϵ for an approximate method, it is easy to see that the function evaluation can be located anywhere inside a distortion sphere of radius D centered around point f without changing the error bound of the approximation technique. In practice, however, the function evaluation f is not known, but f_A is known and ϵ can be theoretically estimated. It is possible to get an estimate of the distortion radius D' as:

$$D' = \epsilon |f_A|.$$

It is easy to see that:

$$\frac{|D - D'|}{|D|} < \epsilon.$$

Since ϵ for typical applications is in the range of 10^{-3} or less, the resulting error in distortion radii is small. A second problem arises because the center of the distortion sphere f is not known. Rather, the known point f_A can lie

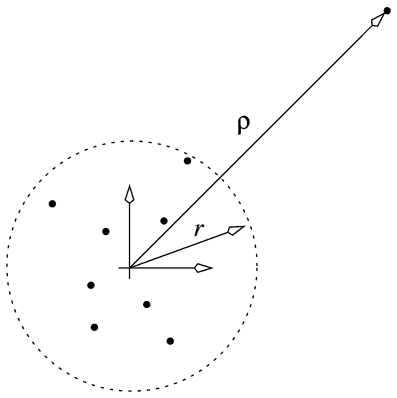


Fig. 1. Error estimation scenario for a single particle—cluster interaction.

anywhere within the distortion sphere. This problem is addressed subsequently in Section 3.2.

2.1 Error in a Single Particle-Cell Interaction

Consider a multipole-based Barnes–Hut method for computing the force and potential on each particle. The basis for this method is the aggregation of a cluster of particles into a single series representation that can be evaluated at various observation points. This is illustrated in Figure 1. The source points are contained within a sphere of radius r and the evaluation point is at a distance ρ from the center of the sphere. The sources are approximated by a truncated multipole series of degree p defined with respect to the center of the sphere. With this as the basic interaction primitive, the following error bounds have been theoretically established:

THEOREM 1 (RELATIVE ERROR IN POTENTIAL USING MULTIPOLES) [GREENGARD AND ROKHLIN 1987; ELLIOTT 1994]. *For a spherical cell of radius r containing a set of charges, the relative error in evaluation of potential at an observation point at distance ρ from the center of the sphere using a multipole series with p terms is given by*:

$$\epsilon_{potential} \leq \left(\frac{r}{\rho}\right)^{p}. \tag{1}$$

PROOF. This error bound is derived in Greengard and Rokhlin [1987] and Elliott [1994]. We refer the interested readers to these detailed derivations. □

THEOREM 2 (RELATIVE ERROR IN FORCE USING MULTIPOLES) [ELLIOTT 1994]. *For a spherical cell of radius r containing a set of charges, the relative error in evaluation of force at an observation point at distance ρ from the center of the sphere using a multipole series with p terms is given by*:

$$\epsilon_{force} \leq \left(\frac{r}{\rho}\right)^{p} \left(p\left(\frac{\rho}{r}+1\right)-1\right). \tag{2}$$

PROOF. This error bound is derived in Elliott [1994]. We refer the interested readers to these detailed derivations. □

2.1.1 *Absolute Error in Potential Using Multipole Expansions.* In addition to relative error, it is often useful to examine the absolute error in potential resulting from a truncated p-term multipole approximation. Absolute error in a function evaluation f approximated by f_A is defined as $|f - f_A|$. The absolute error in potential ϕ resulting from a p-term multipole approximation is given by:

THEOREM 3 (ABSOLUTE ERROR IN POTENTIAL USING MULTIPOLES) [GREENGARD AND ROKHLIN 1987]. *The absolute error δ between the exact potential ϕ and potential ϕ_A computed using a p-term multipole approximation is given by*:

$$\delta = |\phi - \phi_A| \leq \frac{A}{\rho - r}\left(\frac{r}{\rho}\right)^{p+1}, \tag{3}$$

where $A = \sum_{j=1}^{k} |q_j|$ is the sum of magnitudes of all charges contained in the sphere of radius r.

This error metric is often used when absolute error bounds have been specified. Another use of this error bound is in reducing the overall error in simulation. A careful examination of Eq. (3) reveals that the error increases linearly with magnitude of enclosed charge. Consider a uniform charge distribution simulated using a multipole-based Barnes–Hut method. Since a particle interacts with clusters of varying sizes, the larger clusters contribute most to absolute error in evaluated potential. This motivates a variable degree multipole method in which larger clusters (in terms of charge) have a higher degree multipole approximation. The following theorem specifies the variable degree multipole method:

THEOREM 4 (VARIABLE DEGREE MULTIPOLES) [GRAMA ET AL. 2000]. *The polynomial degree p_k required for a particle—cluster interaction for constant absolute error is given by*

$$p_k = p_0 + k \log_\alpha 2 + \log_\alpha \frac{A_0}{A_k}$$

where A_k is the net charge on the cluster at level k and A_0 is the smallest net charge cluster at lowest level in the tree.

PROOF. The increase in polynomial degree is computed by equating the absolute error in a particle—cluster interaction to a constant. Detailed proof is presented in Grama et al. [2000]. □

2.2 Number of Interactions for a Particle in Multipole-Based Barnes–Hut Method

In the Barnes–Hut method, an interaction between a cluster and a particle occurs only when the multipole acceptance criteria, also known as the α criterion is satisfied. This criteria requires that the ratio of the size of the cluster (radius r of circumscribing sphere) to the distance of observation point from center of cluster (ρ) is less than a constant (α, typically in the range 0.5–0.7). This criteria can be used to establish the following observations:

—the number of interactions with clusters of a particular size are bounded by constant, and

—the number of distinct sizes of clusters is equal to the height of the decomposition tree.

For structured distributions with uniform charge density, this translates to $O(\log n)$ interactions. The first assertion can be formally derived as follows:

LEMMA 1 (PERMISSIBLE INTERACTIONS IN BARNES–HUT METHOD) [GRAMA ET AL. 1998]. *In the Barnes–Hut method, the ratio r/ρ for particle—cluster interactions is bounded as follows*:

$$\alpha' < \frac{r}{\rho} < \alpha,$$

where α' and α are constants, such that

$$\alpha' = \left(\frac{2}{\alpha} + \frac{1}{\sqrt{2}}\right)^{-1}.$$

PROOF. The upper bound is established directly by the α criterion and the lower bound is established by using the fact that an interaction with the parent was ruled out by the α criterion. Detailed proof is presented in Grama et al. [2000]. □

As α is reduced, this bound tends to $\alpha/2 < \rho/r < \alpha$, indicating a tight bound. It is now easy to show that the number of interactions with a box of size ρ is bounded by a constant.

LEMMA 2 (BOUNDING INTERACTIONS IN BARNES–HUT METHOD) [GRAMA ET AL. 1998]. *In Barnes–Hut method, a particle interacts with a bounded number of boxes of a given size.*

PROOF. The proof is based on bounding the number of boxes that can be packed into an annular region centered around the observation point within which all boxes of specified size must lie. Detailed proof is presented in Grama et al. [2000]. □

Finally, the total number of interactions can be bounded by the product of the depth of the hierarchy and the number of interactions with a box of each size.

2.3 Aggregate Errors and Time-Step Errors

The errors (distortion radii) introduced into the force/potential/position at each timestep are simply computed as products of magnitude of interaction error and number of interactions for each particle. The distortion radius of a quantity may change as the simulation proceeds over multiple timesteps. For example, for a second order time integration scheme, the error in particle position is given by the following theorem:

THEOREM 5. *After n timesteps of size Δt, the distortion radius for particle position is specified by*

$$|E_n| \leq \frac{T^2 + T\Delta t}{2}\left[c_1 + \Delta t^2 c_2\right],$$

where $T = n\Delta t$ is the total time, and c_1 and c_2 are constants.

PROOF. The proof relies on a simple analysis of a Verlet-Leapfrog scheme to compute the position and velocity of each particle:

$$v_{k+1/2} = v_{k+1/2} + a_k \Delta t, \tag{4}$$

$$s_{k+1} = s_k + v_{k+1/2}\Delta t, \tag{5}$$

where s_k, v_k, and a_k denote the position, velocity, and acceleration, respectively, of a particle at time t_k after k timesteps of size Δt each, that is, $t_k = k\Delta t$. Detailed analysis is presented by Littell et al. [1997]. □

These theoretical results form the basis of our compression framework. After each timestep, the bounds on distortion radius are computed and passed along with the parameters to the compression framework. This framework is described in subsequent sections.

3. FRAMEWORK FOR BOUNDED DISTORTION MULTIDIMENSIONAL QUANTIZATION

In this section, we describe our framework for compressing particle data. We start with a simple scheme, point out inadequacies of this scheme and describe a general purpose approach based on hierarchical decompositions. We improve the performance of this hierarchical decomposition scheme with optimizations such as spatial and temporal coherence. We conclude with a discussion of another scheme based on an index timestep approach, similar to the one taken in MPEG video, that overcomes the drawbacks of hierarchical multidimensional quantization.

3.1 A Simple Scheme for Uniform Quantization

Consider a given two-dimensional domain with particles and associated distortion radii. Instead of representing the particle coordinates using conventional 32-bit floating point numbers, we impose a uniform mesh over the domain. The particles are then moved to the nearest mesh node that falls within the distortion radii of the particle. We illustrate this process for a 3-particle system in Figure 2. The number of mesh points in either dimension is assumed to be a power of two (without loss of generality). With eight discretization points along each dimension, the `x` and `y` coordinates can be represented using 3 bits. The particles `a`, `b`, and `c` can be represented as `(001 010)`, `(110 011)`, and `(101 101)`, respectively.

In the simple case presented above, the scheme manages to reduce the storage requirements from 64 bits (2×32 bits) to 6 bits, a compression factor of over 10. In a general case, if the distortion radius is given by d, the mesh discretization will also have to be of the order of d. The corresponding bit requirement for

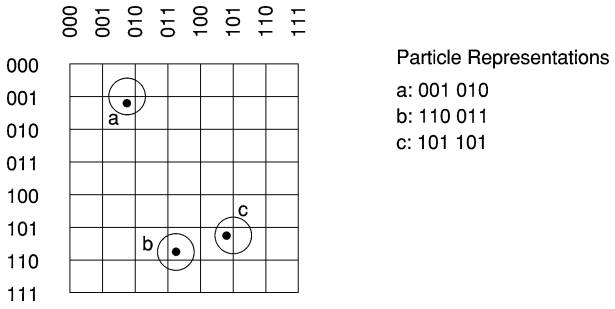


Fig. 2. A simple scheme for quantizing particle positions based on specified distortion radius.

each dimension is given by $-\log_2(d)$. For a 3-D dataset with $d = 10^{-3}$, the storage requirement is 30 bits/particle/timestep. The corresponding compression ratio is roughly equal to three.

While this quantization scheme based on a uniform mesh works well for largely regular particle distributions, its performance can be poor for variable distortion radii or highly irregular distributions. In the case of variable distortion radii, the mesh discretization will have to be refined to the lowest distortion radii of any particle. Note that hierarchical treecodes are capable of computing distortion radii of each particle explicitly in addition to the potential and/or force on the particle. In the case of irregular distributions, fixed length representations such as the one presented above lead to poor compression. In the next section, we present a robust framework that uses the same hierarchical decomposition that is used for force computation by multipole methods. We augment this framework with various optimizations for further improving compression ratios.

3.2 Multidimensional Quantization of Parameter Lists with Arbitrary Distributions

The multidimensional quantization scheme proposed in this paper constructs a hierarchical domain decomposition from a given set of particles and distortion radii. For a two-dimensional problem, the domain is recursively subdivided into quads until each quad contains one particle and the center of the quad is within the distortion radius of the particle contained within. Note that the hierarchical decomposition thus obtained is a refinement of the hierarchy used for force/potential estimation using Barnes–Hut or Fast Multipole Methods. This tree construction process is illustrated in Figure 3. It is clear that this hierarchy may potentially result in a large number of empty leaf-level quads. In our compression scheme, the hierarchy is never explicitly constructed; rather, we simply compute a representation of the required leaf-level quads containing given particles. The multidimensional quantization process can be generalized to any number of dimensions. For storing positions of three-dimensional point-sets, the quantization process results in oct-trees; that is, the domain is recursively subdivided into eight octs at each step.

Once this hierarchical structure has been constructed, particles are assigned to leaf nodes that lie within the distortion radii of the particle. The problem

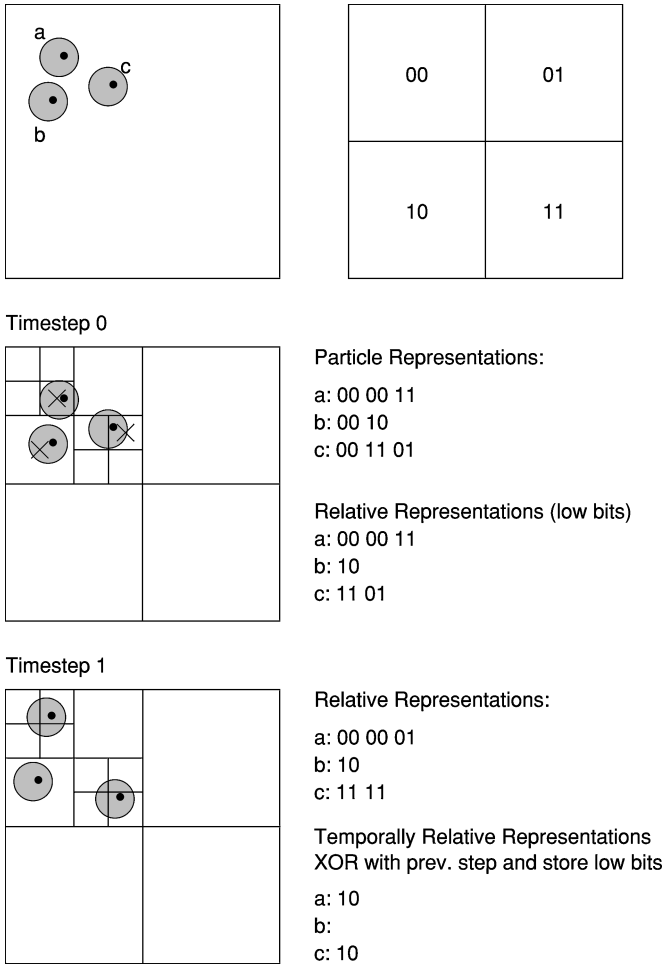


Fig. 3. Illustration of compression of particle position data using a distortion sphere, and spatial and temporal coherence.

of representing particle positions now reduces to the problem of representing populated leaf-level octs (nodes) in the oct-tree. This is done by encoding the path from the root to the leaf node. By associating a predefined ordering of children in the oct tree, we can associate 3 bits per level in the tree. We illustrate this process for a 2-D problem in Figure 3. In this case, we require only 2 bits per node since the tree is a quad tree. In the example, particle a is at level three in the tree and according to the predefined node ordering for children of a node, it is represented as `00 00 11`. This provides the basic quantization mechanism for our schemes.

Discussion. When compressing particles in the above framework, the accurate particle position f is not available to us. Instead, the approximate (distortion bounded) position f_A is available to us. Consequently, if we select d' to be the distortion radius for compression in the above framework, in the worst case,

a particle may be separated from the accurate position by a distance $d + d'$, where d is the original distortion bound induced by the hierarchical treecode. We select $d' = \mu d$, where μ is a constant (typically less than 0.01), for a true distortion bound of $(1 + \mu)d$.

3.3 Encoding Spatial Coherence

The next order of compression relies on spatial coherence of representation. Simply stated, particles that are spatially proximate are likely to share large prefixes in the path from root to leaf. This implies that if the particles are sorted in a proximity preserving order (such as Morton or Hilbert curves), then we can represent particle positions relative to the previous particle. Indeed, for improved cache performance as well as parallel performance, particles are typically sorted in a spatial order (such as a Morton order or a Peano–Hilbert order) for the FMM/Barnes–Hut computation [Grama et al. 1996; Singh et al. 1994; Warren and Salmon 1993], and this requires no additional processing for compression.

The use of spatial coherence for improving compression ratios is illustrated in Figure 3, Timestep 0. The quantized representations for particles `a`, `b`, and `c` are given by `00 00 11`, `00 10`, and `00 11 01`, respectively. Assuming that these particles are sorted in the order `a`, `b`, and then `c`, it is easy to see that particles `b` and `c` share the prefix `00` with particle `a`. Consequently, the prefix does not need to be stored for these and the representations for `b` and `c` are simply `10` and `11 01`. While this may not seem to be a significant improvement in this example, in typical trees, the depth can be high. For example, with a normalized domain of unit size in each dimension, a distortion radius of 10^{-3} would require up to 10 levels in the oct tree. In such cases with higher particle densities, significant improvements result from the use of spatial coherence in our quantization framework.

The performance of spatial coherence encoding is impacted by the fact that two spatially proximate particles may have significantly different prefixes based on which half of the tree they fall into. Trivially, two particles that lie on either side of a half split will have entirely different prefixes. In such cases, the spatial coherence framework does not provide any additional benefits over simple quantized representation. However, the number of such particles is a small fraction of the total number of particles.

3.3.1 *Encoding Temporal Coherence.* In addition to spatial coherence, compression rates can be further improved by considering temporal coherence; that is, a particle is not expected to move significantly over a single timestep. Consequently, instead of storing the entire relative leaf location corresponding to a populated leaf node, we can store only the difference with respect to the previous location. This is stored as a sequence of XOR'ed least significant bits. This is illustrated in Figure 3 (Time-step 1). The quantized positions of particles `a`, `b`, and `c` are given by `00 00 01`, `10`, and `11 11`, respectively after coding spatial coherence. These representations can be reduced to `10` and `10` respectively for `a` and `c` since the rest of the prefix is shared with the previous timestep. Also, since the representation of `b` did not change between the two timesteps, it does

not require any storage. It is easy to see from this simple illustration that the hierarchical framework when combined with spatial and temporal coherence is capable of yielding very high compression rates.

An important aspect that impacts the performance of time-coherence exploitation is the choice of boundary conditions in the simulation. Typical simulations are run with periodic boundary conditions; that is, the simulation domain is infinitely replicated around itself. With such boundary conditions, a particle leaving the domain of simulation re-enters it on the far side. This causes a significant shift in the particle position and for these particles the differentially encoded position requires the same amount of storage as a naive representation of the path from root to the leaf node. However, the number of such particles is small (by the surface to boundary argument in a steady state simulation) and does not significantly degrade the performance of the compression scheme.

3.3.2 *Coding Particle Displacements and Higher Order Differences.* It is possible to encode particle displacements and higher order differences in the same framework as illustrated above. The driving intuition for this is that displacements of particles between timesteps are not expected to change abruptly. To encode particle displacements in the above framework, the refinement criteria for the oct-tree must be modified slightly—a node is subdivided only until its center is within prescribed distortion radii. The condition relating to single particle per leaf is not required. Furthermore, the distortion radius needs to be altered appropriately to maintain the same distortion radii for particle positions as before.

This compression process is illustrated in Figure 4. Between timesteps 0 and 1, particles `a`, `b`, and `c` are displaced by vectors `(0.1, 0.3)`, `(0.05, 0.25)`, and `(0.35, 0.15)`, respectively. These values are now encoded in our quantization framework as before. Since more than one particle can have the same displacement (but not the same position), multiple particles can be assigned to the same leaf node in the hierarchy. The hierarchy corresponding to the displacements is illustrated in Figure 4. The hierarchy is populated with vectors, each corresponding to the displacement of a single particle. Populated leaf nodes in this hierarchy are now encoded using spatial coherence.

An important consideration in this process is the choice of timestep data with respect to which differences are computed. Consider two vectors T_i and T_{i+1} of positions along with their error-bounded quantized counterparts T'_i and T'_{i+1}. To determine T'_{i+1}, it is natural to consider the displacement $D_{i+1} = T_{i+1} - T_i$ and code it in our framework to give D'_{i+1}. On the decoding side, T'_{i+1} is recovered as $T'_{i+1} = T'_i + D'_{i+1}$. A close examination of this process reveals that this leads to unstable error properties. This follows from the following sequence of steps. We know that the following equations hold for bounded distortion radii η and ν.

$$
\begin{aligned}
D_{i+1} &= T_{i+1} - T_i \\
|D'_{i+1} - D_{i+1}| &< \eta \\
|T'_i - T_i| &< \nu \\
T'_{i+1} &= T'_i + D'_{i+1}.
\end{aligned}
$$

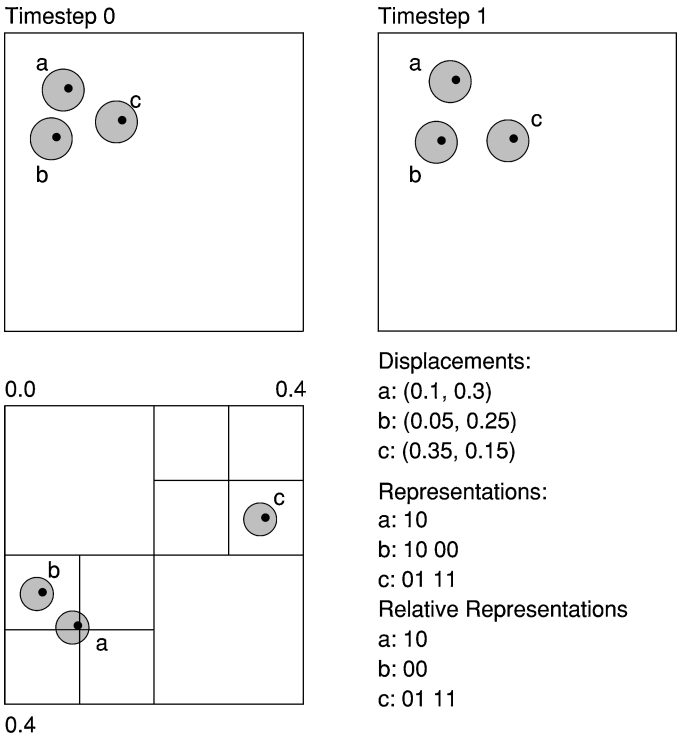


Fig. 4. Illustration of compression of particle displacement data using distortion sphere and spatial coherence. Notice that the extent of the domain represented hierarchically is determined by the maximum displacement in any dimension and that the radius of distortion sphere is in fact smaller than for coding positions.

From these, with some basic algebraic manipulations, it is possible to show that:

$$|T'_{i+1} - T_{i+1}| < \nu + \eta.$$

This is an undesirable situation in which the effective distortion radii for particle positions increases at each timestep.

To remedy this situation, for encoding T_{i+1}, its differences with respect to T'_i must be encoded instead of vector T_i. Mathematically, we have:

$$D_{i+1} = T_{i+1} - T'_i$$
$$|D'_{i+1} - D_{i+1}| < \eta$$
$$|T'_i - T_i| < \nu$$
$$T'_{i+1} = T'_i + D'_{i+1}.$$

In this case, we can show that:

$$|T'_{i+1} - T_{i+1}| < \eta.$$

This is important because we are now able to maintain a constant distortion bound for particle positions. Indeed, this is simple to implement and requires

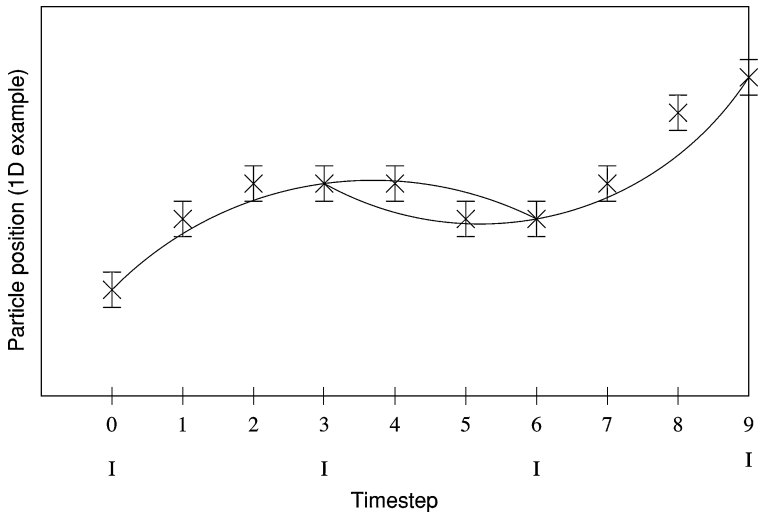


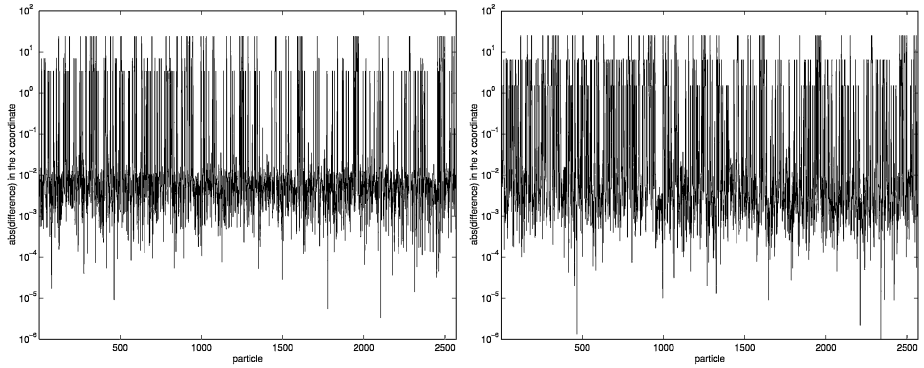
Fig. 5. Using index (I) timesteps for interpolating particle positions for intermediate timesteps.

storing the previous compressed timestep representation while compressing the next timestep. A similar technique is used for coding second-order differences in particle positions.

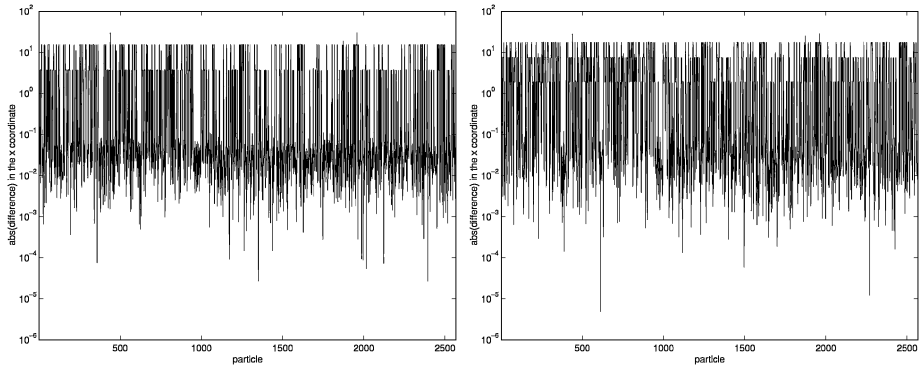
3.4 Index Time-Steps for Interpolated Encoding

One of the major drawbacks of differentially coding timesteps in the above framework is the inability to selectively decompress specified timesteps or regions of space quickly. Specifically, to decompress a certain timestep, the entire simulation leading up to the timestep would have to be unrolled. Furthermore, if a small part of the data was corrupted, the entire data following the corrupted segment would be lost. These drawbacks can be potentially unacceptable in many situations (unreliable media or networks, analysis tasks requiring segmented retrieval). We now propose a scheme that addresses these drawbacks.

An alternate scheme for compressing particle positions relies on using periodic index timesteps (I-steps). Consider the illustration in Figure 5. This example shows a 1D trajectory of a single particle across nine timesteps. Errorbars on particle positions indicate the distortion radii for each particle. In this example, every fourth timestep is an index step (marked by an I in the figure). A simple technique for coding particle positions in this framework uses the neighboring I-steps to construct a polynomial that specifies particle positions at intermediate timesteps. If the polynomial approximation lies within the error-bars, no data needs to be stored for the particle. If the polynomial lies outside the errorbar, then the data can be differentially encoded with respect to the polynomial. In the example in Figure 5, it is easy to see that there is no storage associated with timesteps 1, 2, 5, and 7 (0, 3, 6, and 9 are index timesteps). Only timesteps 4 and 8 require additional encoding. Clearly, this technique has the potential for significant compression.



Magnitude of error in the second interpolated timestep when using two interpolated timesteps between each pair of I-steps.



Magnitude of error in the third interpolated timestep when using four interpolated timesteps between each pair of I-steps.

Fig. 6. Use of I-steps for interpolating intermediate timesteps.

We now look at the reduction in magnitude of quantities that need to be encoded because of the interpolation framework above. In Figure 6, we illustrate the use of I-steps with interpolation polynomials of varying degrees for a 2568 atom system (water molecules). For the sake of illustration, we only show the encoding of the x coordinate. The magnitude of difference between the polynomial and actual position computed from the timestepping scheme applied to the hierarchical method is shown. In the extreme case, if each of the magnitudes in the figures were under, say, 10^{-2}, and the distortion radius was 10^{-2}, then no encoding is required for the intermediate timesteps. In Figure 6, we see that the magnitude of differences can be reduced very significantly by quadratic and cubic interpolation. We experimented with varying number of intermediate timesteps and present results for these in Section 4. Once particle trajectories have been interpolated, the data is differentially encoded with respect to this trajectory. This differential coding is done using the hierarchical framework used for quantizing particle positions as before.

This approach of using I-steps for compression has other benefits over simply coding first and second order differences in particle positions. The I-frame

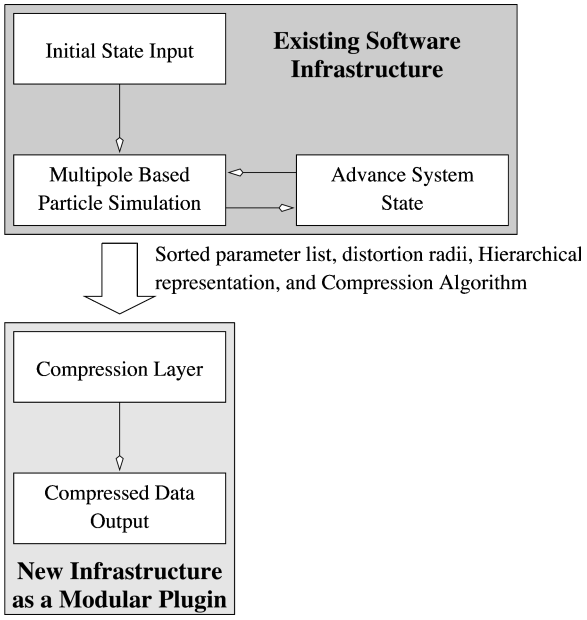


Fig. 7. Organization of major software components and the dataflow among these components.

approach alleviates the drawbacks of differentially coding timesteps and supports fast access to intermediate timestep data with limited unrolling. Note that this approach is similar in philosophy to that taken by MPEG encoders for video compression.

3.5 Software Organization

A high-level overview of various software components is illustrated in Figure 7. The compression layer fits in as a modular plugin into the hierarchical multipole-based treecode. The inputs to this compression layer are:

—A parameter list sorted on the spatial position of the particles. This list consists of n d-dimensional vectors defined over the space of real numbers, where n is the number of particles.

—The distortion radii associated with each particle. This list consists of n real numbers, each specifying the distortion radius associated with corresponding particles in the parameter list.

—The FMM/Barnes–Hut hierarchical representation of the domain. This hierarchical representation is used to accelerate the compression process. The representation can be easily reconstructed in the compression layer as well to make abstract data types more flexible.

—The algorithm to be used for compressing the data and parameters associated with the algorithm where necessary. For instance, while using the I-step approach, the polynomial degree and number of intermediate timesteps between two I-steps are parameters to the compression algorithm.

The compression layer uses the input parameters and stores a compressed representation along with parameters into the compressed output file. In many of the algorithms, timesteps need to be buffered for computing higher order differences or I-step interpolants. These are maintained in static data structures within the compression layer; thus ridding the calling program of additional book keeping. Note that the error-estimation routines for computing distortion radii must still be incorporated into the hierarchical treecode.

3.6 Performance of Compression Schemes

We begin by reiterating that the best compression for such applications is simply to store the initial state of the system. The decompressor is the simulation process itself. This highlights the trade-off between computational overhead (speed of compression and decompression) and the compression ratio. It is important for any compression scheme to have the following properties:

—Fast compression and decompression.

—High compression ratios.

—Data-structures that can build on existing hierarchies and support further analysis tasks on the data.

Our schemes address each of these requirements explicitly. The performance of the proposed schemes is a sensitive function of the distortion radius associated with the quantity being stored. While theoretical bounds exist on distortion radii, it is often observed that these bounds are loose and that hierarchical methods yield much better results in practice. For our compression schemes, we use distortion radii which are much smaller than theoretical bounds. For instance, for a charged particle system with charges totaling to 1, the error behavior of the electrostatic potential is given by α^{p+1}. For $\alpha = 0.5$, an increase in multipole degree by one leads to halving of the error term. In the oct-tree structure for compression, each additional level corresponds to halving of the distortion radius. Therefore, the depth of the tree is expected to grow linearly with the multipole degree. For a compression tree depth of 10, the distortion radius is 2^{-11} and the corresponding storage per path is 30 bits (3 bits per level). Notice that without using any coherence (spatial or temporal), we can already achieve a 3-fold compression (down from 12 bytes) in storing positions. Spatial coherence can be encoded either by sorting particles in a proximity order and coding differences with respect to previous particles or by using tries (as used in LZ compression). We chose to use the former since the particles are already sorted in proximity order for force/potential estimation.

Other issues relating to particle dynamics simulations relate to stability, continuity, and maintaining energy in a closed system. Although our compression framework does not address these issues directly, it maintains stability characteristics of the original (uncompressed) simulation. The I-step approach also provides additional continuity based on the selected interpolation polynomial. These are important orthogonal issues that must be addressed in the simulation itself.

Table I. Number of Particles in Problem Instances, Size of Uncompressed Position Data (Bytes), and Results of LZ Compression

No. of Particles	Uncompressed Size	LZ Compressed Size	Compression Ratio (LZ)
Structured Distributions			
2568	30816	15289	2.02
10K	120K	76123	1.58
20K	240K	157426	1.52
100K	1.2M	704018	1.70
500K	6.0M	3409091	1.76
2M	24M	13114754	1.83
Unstructured Distributions			
24K	288K	241036	1.19
45K	540K	457151	1.18

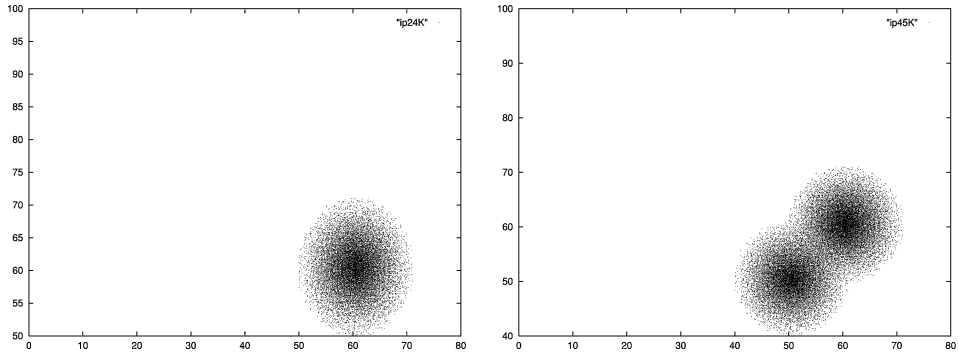


Fig. 8. Sample distributions for experiments: (a) Gaussian (24,000 particles), and (b) Overlapped Gaussians (45,000 particles).

4. EXPERIMENTAL RESULTS

We implemented our compression framework in conjunction with a multipole based Barnes–Hut method with a leapfrog integration scheme. We tested our schemes on various problems with sizes ranging from 2K to 2M particles and distributions ranging from uniform to Gaussian and multiple Gaussians. These distributions are illustrated in Figure 8. Uniform distributions are synthetically generated by locating particles randomly across the domain. Gaussian distributions are generated by altering the point density according to a Gaussian curve and generating the required point density randomly.

The LZ compression (gzip) of particle data forms the baseline for our comparison. In Table I, we present sizes of uncompressed particle data, size of LZ compressed data, and the compression ratio of LZ. Compression ratios for our schemes will be presented as improvements (compression ratios) with respect to LZ compression.

We present results of the basic compression scheme followed by the effect of various optimizations in Table II. The distortion radii used in these cases is presented in Table III. The theoretical distortion radii in each of these cases is over two orders of magnitude more than selected distortion radii. The four schemes presented in Table II are as follows:

Table II. Compression Results for Uniform Particle Distributions of Varying Sizes. Performance of Various Schemes is Computed as Compression Ratio with Respect to LZ-compressed Position Data

Problem Size	Position Spatial Scheme 1	2nd timestep Temporal Scheme 2	Displacement Spatial Scheme 3	2nd Ord. Diff. Spatial Scheme 4
Structured Distributions				
2568	1.44	1.81	1.78	2.01
10K	1.99	3.31	4.19	5.32
20K	2.36	3.45	4.73	6.11
100K	3.20	3.03	5.20	7.12
500K	4.10	4.18	5.06	7.43
2M	4.16	4.09	5.44	7.38
Unstructured Distributions				
24K	4.15	2.24	2.50	3.76
45K	4.98	2.51	2.85	4.27

Table III. Distortion Radius Used for the Particle Sets

	Distortion Radius	
Problem Size	position	difference of position
2568	1.0×10^{-3}	1.0×10^{-4}
10K	3.9×10^{-3}	3.9×10^{-4}
20K	7.8×10^{-3}	7.8×10^{-4}
100K	3.9×10^{-2}	3.9×10^{-3}
500K	4.0×10^{-2}	4.0×10^{-3}
100K	4.0×10^{-2}	4.0×10^{-3}
Non-uniform distributions		
24K	1.0×10^{-2}	1.0×10^{-3}
45K	2.0×10^{-2}	2.0×10^{-3}

Scheme 1 compresses particle positions using only spatial coherence. In this scheme, particles are first sorted in a proximity-preserving order (a space filling curve). The representation of the leaf oct for each particle is coded differentially with respect to the previous particle. This process is illustrated in Figure 3 (Timestep 0).

Scheme 2 compresses particle positions using both spatial and temporal coherence. Particles are first sorted in a proximity preserving order and their representations are coded differentially with respect to previous particle. In addition, these representations are further coded differentially with respect to the (spatially differentially coded) representation for the particle in the prior timestep. This process is illustrated in Figure 3 (Timestep 1).

Scheme 3 computes differences in particle position (displacement) between timesteps and stores them in the spatial coherence framework, that is, particles are sorted in a proximity order and their displacement representations are stored differentially with respect to previous particle. If the timesteps for each particle are identical, then this corresponds to storing velocities of particles.

Scheme 4 computes second-order differences (difference in displacement of particle) and stores them using spatial coherence. If the timesteps for each particle are identical, then this corresponds to storing acceleration for each particle.

The following observations can be made from the table: coding particle position data using the hierarchical distortion radii proposed in the paper with spatial coherence yields up to a factor of three improvement in compression over gzipped raw data. This corresponds to a compression ratio of six over raw data and can be seen to scale very well with problem size. Applying temporal coherence for coding the next timestep yields improvements in the range of 20% over spatial coherence. This is less than expected; however, it is due to the fact that the selected distortion radii is very small. Consequently, particles traverse significant number of leaf octs in the tree-space. The last two columns correspond to storing first and second order differences in particle positions in the hierarchical framework using only spatial coherence. It can be seen that storing second order differences yields the best storage of any scheme and yields compression ratios ranging from 4:1 to 12:1 over raw data and 2:1 to 7:1 over gzipped data.

An interesting observation from the Table II is that compression improves as the number of particles is increased. This is because the error bound of multipole methods grows with the total charge in the system. Thus, the distortion radii increases with increase in number of particles. Consequently, the compression ratio increases. This also points to an important fact that the accuracy of multipole methods needs to be increased (by changing the α parameter or multipole degree) as systems become larger. In this case, we expect the compression ratios to be largely independent of the size of particle system. We base this on the last two observations we make on larger datasets –500K and 2M particles, respectively. We compress these datasets with constant distortion radii of 4×10^{-2}. In these cases, we see that going from 0.5M to 2M particles does not change the performance characteristics of our compression schemes significantly. It is important to note that while simulations of these magnitudes are adequate for most molecular dynamics simulations, experiments in astrophysics now approach 100M—1G particles.

For unstructured distributions, it is evident from Table II that compression ratios are much lower than their structured counterparts. For example, for a 100K particle uniform distribution, the best encoded size is 98820 bytes, whereas for a 45K particle unstructured distribution, it is 91784 bytes. This large difference is a result of the fact that the tree depth in unstructured distributions can be much higher. For this purpose, it may be necessary to use alternate hierarchical data structures. The discrepancy is also a function of the distortion radii in the two cases. In reality, the distortion radii for unstructured distributions is much higher for simple FMM/Barnes–Hut methods.

The relative improvements due to our schemes are illustrated in Figure 9. It is easy to see that our schemes yield excellent compression ratios in a fast and flexible framework.

In Table IV, we present results from compression schemes based on the use of index timesteps. We present encoded results for quadratic and cubic fitting

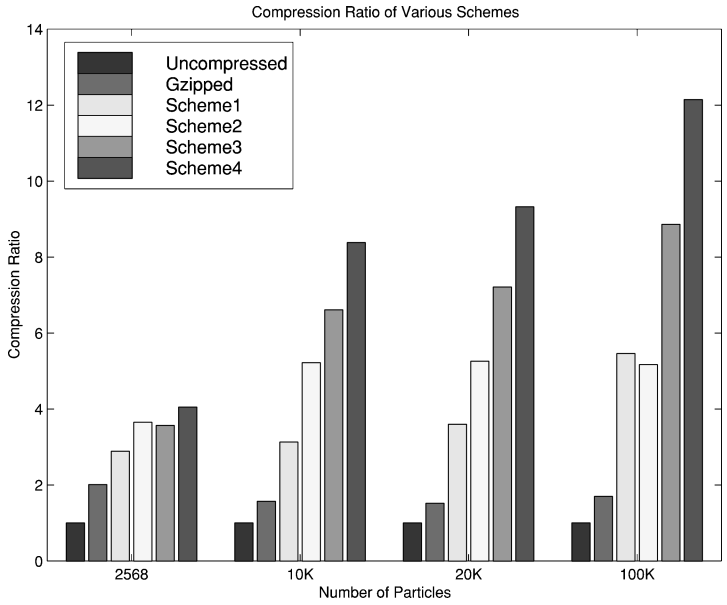


Fig. 9. Comparison of compression rates of various schemes: Scheme 1 compresses particle positions with only spatial coherence, Scheme 2 compresses particle positions using both spatial and temporal coherence, Scheme 3 compresses particle displacements with spatial coherence, and Scheme 4 compresses second order differences with spatial coherence.

for a number of distributions. The size of compressed data is computed as an average of one index timestep and all of the intermediate timesteps up to (but not including) the next timestep. This is done to ensure a fair comparison of various schemes. The number of intermediate (interpolated) timesteps is varied between 2 and 4. Several observations can be made from Table IV and its comparison to Table II. The compression ratio for uniform distributions improves with increasing number of intermediate timesteps. This is expected to saturate and then decrease as the number of intermediate timesteps is further increased. A comparison with Table II also reveals that this scheme yields the best compression of any scheme for smaller systems. For larger systems (100K), the second-order difference scheme with spatial coherence outperforms the I-frame approach. The storage requirement of the scheme is almost linear in the number of particles and approaches 8 bits/particle/timestep. Furthermore, a quadratic fit is largely adequate and higher order interpolations do not result in significant improvements in performance. In some cases, they result in slightly poorer performance. This is because cubic interpolants use the previous four I-steps for prediction and in doing so make potential mistakes.

5. DISCUSSION AND IMPACT

The results presented in this article correspond primarily to storing particle position data although the framework can be applied to other particle parameters as well. In general simulations, one might need to store particle velocity,

Table IV. Compression Results for Particle Distributions of Varying Sizes. N Indicates the Number of Interpolated Timesteps. Compression Ratios are Presented with Respect to LZ-Compressed Position Data

Problem Size	I-step Approach					
	Quadratic Fit			Cubic Fit		
	N = 2	N = 3	N = 4	N = 2	N = 3	N = 4
Structured Distributions						
10K	4.34	5.42	6.32	4.59	5.67	6.36
20K	4.51	5.63	6.48	4.76	5.83	6.45
100K	4.04	4.99	5.55	4.22	5.02	5.25
Unstructured Distributions						
24K	4.33	4.62	4.52	4.29	4.40	4.09
45K	4.66	5.06	5.04	4.64	4.86	4.61

acceleration, potential, force, etc. While one approach is to compute these parameters from particle positions over timesteps, it would involve considerable computational overhead and possibly, numerical inaccuracies. Alternately, we advocate a technique that uses the same hierarchical framework for storing other computed parameters as well. Depending on the timestepping scheme used (e.g., a Leapfrog–Verlet scheme), distortion radii can be computed for particle velocities as well. Distortion radii for gravitational (or electrostatic) potential and force are known directly from error bounds on multipole methods. Using these distortion radii, it is possible to build hierarchical structures for each parameter that needs to be stored independently. Since most particle parameters exhibit spatial coherence (as determined by the particle coordinates), the same (proximity preserving) order can be used to differentially code other parameters as well. Schemes 3 and 4 in Table II correspond to storing velocity and acceleration data in the case of identical timesteps for each particle.

The compression schemes presented in this paper reduce storage requirements of particle data by as much as a factor of 12 while guaranteeing distortion bounds on compressed data. This has several important implications for particle simulations: given a subsampling frequency, the I/O overheads can be reduced significantly; conversely, given a desired overhead factor, the sampling rate can be increased significantly for better analysis. It also facilitates remote access across networks where bandwidth is a bottleneck. The framework put forth in this article provides a natural mechanism for clustering and quantifying cluster behavior. It can be used to identify high-energy regions of the domain and other quantitative subdomain features. It also supports segmented retrieval without expensive unrolling overheads in asymptotically optimal time.

The compression algorithm can be easily parallelized (threaded) and incurs little parallel overhead. Subdomains assigned to processors can be independently compressed with no serialization or communication overheads. This also provides a natural declustering of particle data across disks for improved parallel I/O performance. There is a slight loss of compression in the parallel context since each processor treats its subdomain independently of

the others; thus leading to loss of coherence. However, in practice, this loss of compression is expected to be negligible.

We are currently exploring techniques for integrating our compression framework with advanced analysis tools and for supporting progressive visualization and transmission of large datasets. Publications and software related to this paper are available at `http://www.cs.purdue.edu/homes/ayg/PARTICLES/`.

ACKNOWLEDGMENTS

The authors would like to acknowledge the referees for making several insightful and encouraging comments, which helped in improving this manuscript significantly.

REFERENCES

ALURU, S. 1996. Greengard's n-body algorithm is not $O(n)$. *SIAM J. Sci. Comput. 17*, 3, 773–776.

BAJAJ, C., PASCUCCI, V., AND ZHUANG, G. 1999. Single resolution compression of arbitrary triangular meshes with properties. In *Proceedings of Data Compression Conference*.

BARNES, J. AND HUT, P. 1986. A heirarchical $o(n \log n)$ force calculation algorithm. *Nature*, 324.

CALLAHAN, P. B. AND KOSARAJU, S. R. 1992. A decomposition of multi-dimensional .point-sets with applications to k-nearest-neighbors and n-body potential fields. In *Proceedings of 24th Annual ACM Symposium on Theory of Computing* (May). ACM, New York, pp. 546–556.

DEERING, M. 1995. Geometry compression. In *Proceedings of SIGGRAPH' 95*. ACM, New York, pp. 13–20.

ELLIOTT, W. 1994. Multipole algorithms for molecular dynamics simulation on high-perfomance computers. Dept. Electrical Engineering, Duke Univ.

GRAMA, A., KUMAR, V., AND SAMEH, A. 1996. Parallel hierarchical solvers and preconditioners for boundary element methods. In *Proceedings of the Supercomputing Conference* (Pittsburgh, Pa.).

GRAMA, A., SARIN, V., AND SAMEH, A. 2000. Improving error bounds for multipole-based treecodes. *SIAM J. Sci. Comput. 21*, 5 (May), 1790–1803.

GREENGARD, L. AND ROKHLIN, V. 1987. A fast algorithm for particle simulations. *J. Comput. Phys. 73*, 325–348.

LEVOY, M. 1995. Polygon-assisted jpeg and mpeg compression of synthetic images. In *Proceedings of SIGGRAPH' 95*. ACM, New York, pp. 21–25.

LITTELL, T. R., SKEEL, R. D., AND ZHANG, M. 1997. Error analysis of symplectic multiple time stepping. *SIAM J. Numer. Anal. 34*, 5, 1792–1807.

SINGH, J., HOLT, C., TOTSUKA, T., GUPTA, A., AND HENNESSY, J. 1994. Load balancing and data locality in hierarchical n-body methods. *J. Parall. Dist. Comput.*

TAUBIN, G. AND ROSSIGNAC, J. 1998. Geometric compression through topological surgery. *ACM Trans. Graph. 17*, 2, 84–115.

WARREN, M. AND SALMON, J. 1993. A parallel hashed oct tree n-body algorithm. In *Proceedings of Supercomputing Conference*.

ZIV, J. AND LEMPEL, A. 1977. A universal algorithm for sequential data compression. *IEEE Trans. Inf. Theory 23*, 3, 337–343.

ZIV, J., AND LEMPEL, A. 1978. Compression of individual sequences via variable-rate coding. *IEEE Trans. Inf. Theory 24*, 530–536.

Received October 2000; accepted July 2001