Distribution-based Particle Data Reduction for In-situ Analysis and Visualization of Large-scale N-body Cosmological Simulations

Guan Li[*1,2] Jiayi Xu[†3] Tianchi Zhang[‡4,2] Guihua Shan [§1] Han-Wei Shen [¶3]
Ko-Chih Wang [‖5] Shihong Liao [**4] Zhonghua Lu[††1]

[1] Computer Network Information Center, Chinese Academy of Sciences
[2] University of Chinese Academy of Sciences
[3] The Ohio State University
[4] National Astronomical Observatories, Chinese Academy of Sciences
[5] National Taiwan Normal University

2020 IEEE Pacific Visualization Symposium (PacificVis) 978-1-7281-5697-2/20/$31.00 ©2020 IEEE 10.1109/PacificVis48177.2020.1186

ABSTRACT

Cosmological N-body simulation is an important tool for scientists to study the evolution of the universe. With the increase of computing power, billions of particles of high space-time fidelity can be simulated by supercomputers. However, limited computer storage can only hold a small subset of the simulation output for analysis, which makes the understanding of the underlying cosmological phenomena difficult. To alleviate the problem, we design an in-situ data reduction method for large-scale unstructured particle data. During the data generation phase, we use a combined k-dimensional partitioning and Gaussian mixture model approach to reduce the data by utilizing probability distributions. We offer a model evaluation criterion to examine the quality of the probabilistic distribution models, which allows us to identify and improve low-quality models. After the in-situ processing, the particle data size is greatly reduced, which satisfies the requirements from the domain experts. By comparing the astronomical attributes and visualizations of the reconstructed data with the raw data, we demonstrate the effectiveness of our in-situ particle data reduction technique.

Index Terms: Human-centered computing—Visualization—Visualization application domains—Scientific visualization;

1 INTRODUCTION

In the past decades, a wide range of observations in astronomy have pointed to a concordance cosmological model, i.e., our Universe is made of 4.86% ordinary matter (or baryonic matter), 25.89% cold dark matter, and 69.11% dark energy [6]. According to today's standard galaxy formation framework, dark matter plays an essential role in the formation of large-scale structures (e.g., dark matter halos) in which the observed luminous galaxies reside. However, the formation of dark matter structures is a highly non-linear process. Currently the large-scale cosmological N-body simulation works well to give a precise estimation. Hence, to reach cosmic analysis with a high precision, cosmological simulations data are essential for theoretical predictions so that we can use galaxy observations to constrain our theoretical models. Starting from the 1970s, the scale of cosmological N-body simulations increases significantly. The increasing number of particles used in simulations approximately matches the empirical 'Moore's law', i.e., the number of particles is doubled every 16.5 months [27]. The current state-of-the-art cosmological N-body simulations can process around 2 trillion particles, requiring about 22 terabytes to store the position information of the particle for each time step [23]. Due to the limitation of storage resources, it is impossible to save all the required simulation data for analysis and visualization. Domain experts often use a few key snapshots and some statistical results. But with those limited data, it is difficult to understand the evolution of the large-scale structure of the universe. Domain experts need to save enough temporal snapshots of the positional information to study the evolution of large-scale structures in the universe. They desire that these data be used for data analysis, estimation of cosmic physical properties, and high-resolution visualizations.

In this work, we reduce cosmological simulation data in-situ to address the challenge of limited I/O bandwidth and computer storage. In-situ data reduction, i.e., in-place compression of the data when it is generated in the calculation process, combines simulation calculation with data reduction processing together. As a result, the amount of data that needs to be saved and transmitted will be greatly reduced. Using probability distributions to process the data has benefited many visualization tasks, especially for data reduction [9, 11]. We present an in-situ data reduction method to compress large-scale particle data by using probability distribution functions.

During the simulation stage, the distribution of the cosmological particle data is irregular and unpredictable in each compute nodes. One of the challenges is that it is impossible to use complex operations to group the particle data with limited computer resources and processing time. This requires us to find an effective way to group the particles. To use the probability functions to estimate the particle attributes, the Gaussian mixture model [16] is used. Although the more probability distribution function models are used the smaller errors will be incurred, more storage will be required as a result. At the same time, the number of Gaussian components determines the quality of the model. Furthermore, the data content that needs to be saved is also different for the different application requirements. Therefore, another challenge is how to choose the appropriate parameters for different applications and requirements. The traditional compression method takes error bounds as the criterion. It calculates the errors of the whole data by checking the value of single floating-point number. Using this criterion can guarantee the quality of data, but it limits the compression rate of data. The cosmological simulation focuses on the feature structures of the particle distribution rather than on a single particle. Therefore, the last challenge is how to quantitatively analyze the error of the cosmological feature structure of reconstructed data.

To address those challenges, we propose an in-situ data reduction framework for N-body cosmological simulations. We used a two-

*e-mail: liguan@sccas.cn
†e-mail: xu.2205@osu.edu
‡e-mail: tczhang@nao.cas.cn
§e-mail: sgh@sccas.cn (corresponding author)
¶e-mail: shen.94@osu.edu
‖e-mail: kcwang@ntnu.edu.tw
**e-mail: shliao@bao.ac.cn
††e-mail: zhlu@cnic.cn

2020 IEEE Pacific Visualization Symposium (PacificVis)
978-1-7281-5697-2/20/$31.00 ©2020 IEEE

stage space partitioning method to establish the correspondence between the distribution model and the cosmological particle data. The k-d tree [3] is used to divide the space in the first step of space partitioning. After partitioning, we are using the Gaussian mixture models to fit the particle attributes. In the second step, we evaluate the quality of the model and divide the space further if the model is of low quality. The I/O storage is greatly reduced by our distribution-based method. To select the number of Gaussian components, we propose an information theory selection method. In the post-data processing stage, we reconstruct particle data from the Gaussian mixture models. We verify the quality of the data by calculating the physical properties of the universe including power spectrum and halo mass function. Using the physical properties as the criteria allows domain experts to evaluate the compressed data from different aspects for scientific applications. In summary, the contributions of this work are threefold:

- An in-situ N-body cosmological simulation data processing framework for analysis and visualization.
- A method of selecting parameters in the Gaussian mixture model for different requirements and a way to establish the correspondence between the model and cosmological particle data.
- New ways for evaluation based on physical property for cosmological particle data.

2 Related Work

Data reduction: Due to the limitation of the storage, it is hard to save all raw data of large-scale data simulations. Traditional methods such as storing key time steps and decreasing the precision of data can significantly reduce the data size [21, 28]. But, the results of these methods may lose important information. Data compression is widely studied by many researchers. Binotto et al. [4] presented a compression method which uses 3D textures to store the scheme of sparse 4D functions. Guthe et al. [14] presented a compression-based multi-resolution rendering method which allows users to interact with the large volume data. Lum et al. [19] used a hardware-assisted rendering method to compress the data for interactive visualizations. He et al. [15] used distributions to model data and a Bayesian approach to predict streamlines. Chen et al. [5] proposed a method that uses quadratic Bezier curves to model vectors in time-varying data with the low temporal sampling rate and uses a Gaussian model to quantify and refine the reconstructed path lines. Li and Shen [17] proposed a framework which uses Gaussian mixture models to model vectors in a data sub-block and use winding angle distributions to maintain the quality of reconstructed streamline quality. Wei et al. [31] represented data by small number of samples which are drawn using importance sampling.

In-situ techniques: Using in-situ processing technology can prevent storing large amounts of raw data. Ma et al. [20] discussed the current problems and future applications of in-situ technology; he also pointed out that in-situ visualization is an effective way to analyze large-scale simulations. Many studies have achieved good results through in-situ techniques. Yu et al. [35] used in-situ visualization techniques to analyze combustion data and achieved high-quality visualization results. These visualization results help domain experts to better understand the highly intermittent transient phenomena. Ahrens et al. [1] rendered images in-situ and greatly reduced data storage. Their approach saves images to allow scientists to explore data interactively. Wang et al. [30] proposed a compact view-dependent data proxy. Their technique produces the data proxy in-situ which allows users to render the data with different transfer functions. Vishwanath et al. used GLEAN [29] which is an in-situ framework to enhance the data analysis pipeline. Woodring et al. [32] used a shared mesh data structure for in-situ analysis, which can greatly reduce resource usage. Woodring et al. [34] used in-situ workflow in an ocean simulation to greatly enhance the ability of data processing and analysis. Dutta et al. [9, 10] used Gaussian mixture model to estimate jet engine simulation data in-situ, which can significantly reduce the storage. They used the model to statistically detect anomaly in the post-analysis stage and help scientists study the spatiotemporal trends of rotating stall. For the cosmological simulations, Woodring et al. [33] proposed a random in-situ sampling method to reduce the saved data. By using the sampling-based method, they can help domain experts analyze the entire particle population and interactively visualize the data. But, when the compression rate is high, the method for finding haloes in low-resolution still needs to be improved.

3 Background

The large-scale spatial distribution of matter in the universe is like a "cosmic web", which is composed of halos, filaments, sheets and voids. These structures originate from an initial non-uniform density distribution at the early stage of the universe and affect its later evolution. Therefore, studying the properties of the cosmic web in simulations and comparing them with astronomical observations are essential for cosmological scientists to understand the history of the universe and constrain current cosmological models.

The GADGET-2 simulation program is an open source software for cosmological N-body simulations [26]. In this work, we use the GADGET-2 to simulate the temporal movement of dark matter particles. In the initial stage of simulation, the particles are randomly distributed in the space. Then the particles evolve to form voids, sheets, filaments, halos and other cosmic structural features under the force of gravity. Each particle has seven attributes including the 3D spatial coordinates, the 3D velocity vector, and the particle id. The spatial distribution of particles also can be converted to the density field. Domain experts can study the formation process of cosmic structure features by analyzing the spatial distribution of particle density at higher temporal resolution.

If more particles are used in the simulation program, the simulation results are more accurate. However, large-scale simulation requires a huge amount of computing power and storage. Moreover, scientists need to use different physical parameters to simulate the universe many times, and then validate the scientific hypothesis by analyzing the resulting ensemble data of different parameters. Due to the storage limitation, domain experts can not store enough snapshots from the large-scale simulation which makes it hard for them to analyze and visualize the evolution of the universe. They hope to further understand the process of the large-scale evolution of the universe, that is, to analyze the distribution of particle data in simulation space at higher temporal resolution. The following requirements from the domain experts are identified:

- Significantly reduce the size of saved data but produce enough snapshots.
- Display high-resolution visualizations, clearly show the universe structures of halo, sheet, filament and void.
- Support necessary astronomical attributes calculation such as the power spectrum and the halo mass function from the saved data. The errors for the computed attributes need be kept small.

4 Overview

We solve the requirements of the domain experts by a distribution-based method for in-situ data reduction. Our goal is to reduce data size for astronomical particles in-situ to satisfy the compression ratio which is defined by the domain expert. Furthermore, our method should be highly scalable to support the large-scale parallel cosmological simulation. We added our in-situ data reduction program to the GADGET-2 simulation program. After combining the module,

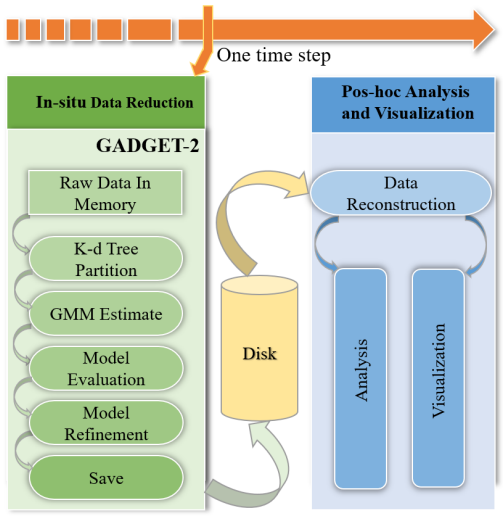


Figure 1: The high-level pipeline of our method.

our in-situ data reduction program runs simultaneously with the simulation program. A high-level overview of the pipeline is shown in Figure 1.

The GADGET-2 simulation program uses multiple processors. The particles are distributed on different processors, and the particle data in each processor is stored in memory using a big array. Since the number of particles is very large, we use the k-dimensional tree (k-d tree) to split the particle data in each processor. The particles can be partitioned quickly and the structure of the partitioning can be maintained by using the k-d tree algorithm. After the partitioning, the leaf node of the k-d tree stores particles in a sub-space. In order to approximate the particles in each leaf node of the k-d tree, we use the Gaussian mixture models (GMM) to compress the particle data in each leaf node. The number of components in the GMM comes from an estimation which is decided by running an experiment on a small dataset. We define a model evaluation formula to evaluate the quality of the GMM. This allows us to identify models that have insufficient quality and to refine them for quality improvement. After enhancing low-quality models, we can achieve the quality requirement for reconstruction and the expert-defined compression rate. At this point, we will stop processing and save the relevant model information to disk. In the data reconstruction stage, to satisfy the requirement of domain experts in calculating astronomical attributes, we reconstruct the particle data. After reconstruction, We can verify the quality of the data by calculating the physical properties. Also, the reconstructed data can be used for visualization.

5 In-situ Data Reduction

In this section, we discuss the details of our distribution-based in-situ data reduction method. It includes space partitioning, data modeling of attributes of particles in each sub-space, model quality evaluation, and the selection of Gaussian components. At the end of this section, we will describe what we store for post data analysis and visualization.

5.1 K-D Tree based Two-stage Space Partitioning

The GADGET-2 simulation program divides the space according to the number of processors to ensure load balancing. Different processors have non-overlapping sub-spaces. However, after the pre-partitioning of the GADGET-2 simulation program, each processor still needs to handle many particles. In the in-situ stage, the available resources for data processing are very limited. Therefore, we may not afford numerous calculations to process the particle data. To further reduce the size of data, we use the GMMs to learn the probability distribution functions of the information attached to particles. Different from volume data and point cloud, cosmological particle data have an irregular spatial distribution and have cosmological features with various internal structures. The cosmological particles, as unstructured data, move under the force of gravity where the particles form clusters in various locations of the simulation space. Hence, the particles are highly unbalanced in space.

In order to balance the number of particles in each sub-space before performing data reduction, we propose a two-stage k-d tree partitioning algorithm to subdivide the simulation space. K-d tree is a data structure backed by k-dimensional space subdivision algorithms. It has high efficiency and can store the partition structure with balanced particle counts. We represent the attributes of particles of each leaf node using a multi-dimensional distribution-based model. The total number of models after two-stage partitioning is proportional to the total number (F) of leaf nodes in the k-d tree. Given a compression rate, the size of a model and raw data size, the desired total number (F) of leaf nodes of the k-d tree is calculated by Equation 1.

$$F = \frac{S * r}{P} \tag{1}$$

where P is the required size for storing a leaf node, S is the size of the raw data, r is the user desired compression ratio.

The first stage uses the aforementioned k-d tree partitioning algorithm to subdivide the space to $F * T$ sub-spaces (leaf nodes), where $T(< 1)$ is a user-defined parameter to determine how many sub-spaces are created in the first stage. We quickly create intermediate k-d tree nodes by only considering the balance of particle numbers when subdividing a sub-space. Domain experts filter and search simulation data through location attributes. The partition we used is based on location attributes which can support for location-based data analysis. For each step of the k-d tree construction, we select the leaf node with the space that contains the most number of particles for splitting. We split the space of this selected node into two sub-spaces by cutting particle location attributes. The largest variance of particle location is the splitting dimension. To efficiently subdivide the particles into two subsets, we use the mean value of the splitting dimension to divide the dataset. After the k-d tree subdivision, each leaf node will correspond to a sub-space that contains a subset of the particles, and we use a distribution model to fit the attributes of the particles.

After the first stage, the quality of the distribution model in every sub-space is not guaranteed. Therefore, a second stage, which partitions space by considering model quality, is needed. In the first stage, we subdivide the space to $F * T$ (T belongs to (0, 1)) sub-spaces. That means we have some leeway to deal with low-quality models to maximize quality. The second stage will keep subdividing the space until the total number of sub-spaces reaches F. The difference from the first stage is the criteria to select a sub-space for subdivision. Instead of choosing the sub-space with the most number of particles, this stage considers the quality of resulting distribution models. That is, we pick the spaces of leaf nodes that have lower modeling quality for subdivision and refinement. The details of model quality evaluation will be described in Section 5.2.2. The second stage allows us to refine the lower quality model to reduce errors. Due to the time overheads of additional model evaluation, the second stage is more time-consuming than the first one. So, the selection of T is a trade-off between the in-situ computation time and data representation quality. A smaller T value results in more leaf nodes that are generated in the second stage; although the computation time is longer by using a smaller T, the model quality can be better.

5.2 Distribution-based Particle Attributes Modeling

We use the GMM to represent the attributes of particles in each sub-space. The details of evaluating the distribution representation quality and the data structure stored for post-analysis are described in the following.

5.2.1 Gaussian Mixture Model

Once the aforementioned simulation space partitioning is done, particles will be assigned to leaf nodes, and a multi-dimensional distribution model is used to summarizes the attributes of particles for each leaf node. The distribution representation for the multi-dimensional attributes of particles should have a good trade-off between the storage size and data modeling quality and support the particle-based data analysis and visualization in the post data processing stage. Kernel Density Estimation (KDE) [25] is one of popular non-parametric distribution-based models, but it requires high storage and computational cost to get high-quality. The histogram is another common representation that can be computed quickly, but it needs high storage when representing a multi-dimensional distribution. For parametric distributions, Gaussian-based methods are commonly used, and can compactly represent a distribution because only a few parameters have to be stored. However, when the data distribution is complex, the GMM is more suitable. A GMM uses K Gaussian models to approximate a data distribution. It has high storage efficiency and does not need prior assumptions about the distribution. Some related studies have pointed out that the use of GMM is an efficient statistical data summarization [9, 18]. Our work uses GMM to approximate the distribution of the attributes of particles in each leaf node. Equation 2 shows the formula of a GMM.

$$p_\theta(x) = \sum_{i=1}^{K} \omega_i * \mathcal{N}(\mu_i, \sigma_i) \tag{2}$$

where K is the number of Gaussian components in the GMM and θ represents GMM parameters which consist of weights, means and standards deviations of Gaussian components. ω_i, μ_i and σ_i are the weight, mean and standard deviation for the i^{th} Gaussian component, respectively. The sum of weights must be equal to 1.

To estimate the parameters of a GMM, the Expectation Maximization (EM) [7] algorithm usually is used. The EM algorithm interactively estimates the parameters of a model which can maximize the likelihood of the given data samples. We apply the EM algorithm on particles of each leaf node of the k-d tree to calculate the parameters of a multi-dimensional GMM. Finally, we store the GMMs to represent the particles in the simulation space. The number of components is an important parameter that influences the storage size and quality when using GMM. We will discuss how to select the number of Gaussian components through the information theory in Section 5.3.

5.2.2 Model Quality Evaluation and Refinement

When using the EM algorithm to model the attributes of particles, the EM algorithm makes the model converged to a local maximum of the likelihood function, but the distribution representation quality is not guaranteed to be good enough. As described in the second stage of the simulation space partitioning algorithm in Section 5.1, our approach improves the quality of data modeling by identifying the GMMs that have low representation quality and then subdividing the simulation space to refine the models with multiple GMMs. Thus, we need a criterion to assess the representation quality. Equation 3 shows the criterion which is the likelihood function of particles and a GMM used to represent the particles.

$$\mathscr{L}(\theta|x_1, x_2, ..., x_N) = \prod_{j=1}^{N} p_\theta(x_j) \tag{3}$$

where x_j is a location of a particle, N is the particle count, $p_\theta(x_j)$ is the likelihood of x_j for a GMM whose parameters are θ. A larger likelihood, $\mathscr{L}(\theta|x_1, x_2, ..., x_N)$, indicates a GMM with a better representation quality. The likelihood function is plausible to evaluate the representation quality of a given model.

Due to the numbers of particles used to calculate GMMs are different, we have to normalize the $\mathscr{L}(\theta|x_1, x_2, ..., x_N)$ calculated from every GMM. Equation 4 is the evaluation formula after the normalization.

$$Score = \frac{Ln(\mathscr{L}(\theta|x_1, x_2, ..., x_n))}{N} \tag{4}$$

where N is the number of particles. *Score* is non-positive and a larger *Score* indicates better quality. In this way, we evaluate the GMMs and identify the models with lower quality to have a higher priority to refine.

With the method to evaluate the representation quality of a GMM, we can determine which models in the sub-space to refine. We sort the leaf nodes according to the corresponding *Score* values. The leaf node which has the smallest *Score* value will be selected and split. Besides, if the storage cost of storing particles of a leaf node directly is less than storing the leaf node represented by the distribution and other information, we will directly store the particle data instead of selecting the sub-space to subdivide. This process is repeated until the total number of leaf nodes is greater than or equal to the desired number (F). After this, we finish the in-situ data reduction for the particle data.

5.3 Number of Gaussian Components

The GMM is composed of several Gaussian components. The number of Gaussian components affects the error and storage of the distribution model. For different applications, the best number of Gaussian components are different. To select the proper number of Gaussian components of a GMM, the Akaike Information Criterion (AIC) is a commonly used estimator [22]. Equation 5 shows the fundamental AIC formula.

$$AIC = 2k - 2ln(\mathscr{L}(\theta|x)) \tag{5}$$

where k is the number of parameters in the GMM. $\mathscr{L}(\theta|x))$ is the value of the maximum likelihood function. A lower *AIC* value means a better modeling quality.

In our algorithm, we have to select a proper number of Gaussian components used by all GMMs to model the whole dataset. To achieve this, we extend Equation 5 to Equation 6.

$$eAIC = 2k * n - \sum_{i=1}^{n} 2ln(\mathscr{L}_i(\theta|x)) \tag{6}$$

where k is the number of parameters of GMMs, n is the total number of GMMs used to model the data. $\mathscr{L}_i(\theta|x)$ is the value of the maximum likelihood function for the i^{th} GMM. Note that n is not equal to the total number of leaf nodes in the k-d tree because some leaf nodes directly store the particles if the storage cost is less than storing a GMM.

For different applications, we first calculate the storage size of a data proxy for different numbers of Gaussian components. The relationship between the number of components and storage size of a leaf node can be described by Equation 7.

$$P = a + b * K \tag{7}$$

a is a fixed number, which is the size to store the total number of particles, the mean particle location, and the maximum distance to mean particle location. b is also a fixed number, which is the size to store one Gaussian component. K is the number of Gaussian

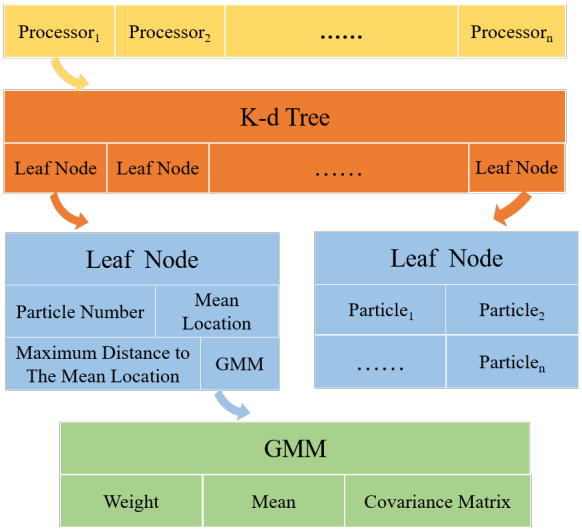


Figure 2: The structure of our data proxy. For the leaf node with a small number of particles, we save the particle attributes directly. Otherwise, we save the particle number, the mean location, the maximum distance to mean location and the GMM parameters.

components. By Equation 1 and Equation 7, we can calculate the desired number (F) of leaf nodes when different Gaussian component counts are given. With those parameters, we can run our data reduction algorithm for the dataset. Thus, the *eAIC* value of the different number of Gaussian components on the dataset can be calculated. The lowest *eAIC* value means the most suitable number of Gaussian components for the application.

In a GMM, the representation quality is better if the number of Gaussian components is larger. However, increasing the number of Gaussian components will decrease the leaf nodes and increase particles in each leaf node, and the representation quality may not significantly improve. Choosing the proper number of components in GMM allows the model to achieve an efficient performance.

5.4 Data Proxy Structure

We introduce the data structure of our data proxy in this section. As described in Section 5.2.2, our data proxy has two types of leaf nodes. One is particle attributes, and the other one is the parameters. The data proxy structure in our application are illustrated in Figure 2. Each processor maintains a k-d tree which partitions a simulation space. When the in-situ data process is done, each process traverses the tree and outputs leaf nodes to disk. The output is stored in two files, a particle file, and a distribution file. For the leaf nodes directly to store particle attributes, these data are copied to the particle file. For the leaf nodes that are represented by the GMMs, we store the particle count, the mean location, the maximum distance to the mean location, and the GMM parameters to the distribution file. The particle counts in the distribution tell us how many particles in the region modeled by a GMM. The mean particle location and the maximum distance to the mean location define a precise spatial region which includes all particles of the leaf node.

6 Data Reconstruction

Our domain experts require that our method supports the analysis of astronomical attributes. The data format that their analysis tools accept is particle data. This requires us to reconstruct the particle data from GMMs to satisfy the requirements of the domain experts.

Our data proxy stores either particles directly or a GMM to represent a leaf node. If a leaf node directly stores particles, we just have to copy particles from the data proxy to memory as the result

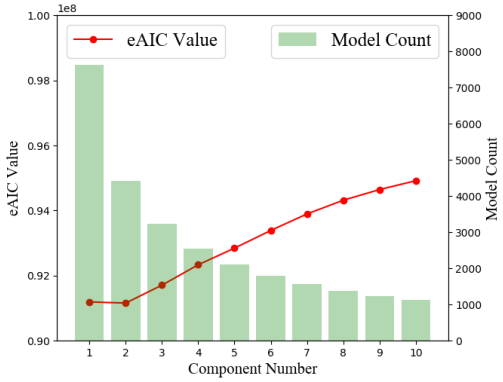


Figure 3: This figure shows the eAIC value and the model counts for different component numbers at the fixed compression rate. The left Y-axis corresponds to the eAIC value. The right Y-axis corresponds to the model count. When the compression ratio is fixed, the model count and the number of components are in inverse proportion.

of reconstruction for the reconstructed data. To reconstruct particles of a leaf node represented by a GMM, we employ the Monte Carlo method for reconstructing particles, as was used in [13, 18]. After reconstructing all leaf nodes, the reconstructed particle data can be simply used by the domain experts in their analysis tasks.

Besides, domain experts often need to analyze and visualize some specific structures. In this scenario, they determine the spatial region of the target cosmic structure based on the statistical information of the particles. A traditional practice to access particles in the region requires traversing all particle data files and finding those in the target region.

This traditional approach results in a huge file I/O overhead to search for a small number of particles. In our approach, because partitioning is based on location attributes and we store the mean particle location and the maximum distance to the mean location for the leaf node, it allows us to define a sphere that contains all particles of the leaf node. When a target region is given, our technique does not have to reconstruct particles of all leaf nodes. We can efficiently check whether the target space intersects with the sphere of the leaf node. If they intersect with each other, we reconstruct particles in the leaf. Otherwise, we simply discard the leaf node. This local data reconstruction scheme can significantly reduce the I/O cost operation and save time for reconstructing particles from the GMMs.

7 Experiment and Evaluation

Given that galaxy velocities are still difficult to observe and hard to model (e.g. redshift space distortion), most of today's observables in cosmology (e.g. matter power spectra, two-point correlation function) actually come from the spatial information. The spatial information of dark matter particles and haloes are very useful in halo abundance matching, cosmic web classification, etc. Domain experts require us to process the location information of particles at the in-situ process stage to support their study. They hope that the size of the saved particle location data is approximately $1/200$ of the full data which contains particle position information, velocity information, and id. Therefore, the compression rate for the position information is 1.33%. In this section, we qualitative and quantitative analyze the performance of our algorithm and the errors between the reconstructed particle data and the raw data.

7.1 Experimental Parameters

The simulation adopts the standard cosmological model with a total matter fraction of $\Omega_m = 0.3089$, a baryonic matter fraction of $\Omega_b =$

0.0486, a dark energy fraction of $\Omega_\Lambda = 0.6911$, a Hubble parameter of $H_0 = 100h$ km/s/Mpc with the dimensionless constant $h = 0.6774$, a power spectrum normalization of $\sigma_8 = 0.8159$, and a primordial power spectrum index of $n_s = 0.9667$ based on the study [6]. The simulation starts at redshift $z = 127$ when the universe was 11.4 million years old, and saves snapshots at every 0.1 gigayear. Here, redshift is a quantity which is widely adopted in astrophysics to denote the cosmic time, with $z = 0$ being the present and $z = \infty$ being the beginning of the universe (i.e., the Big Bang). The relation between the age of the universe and redshift is

$$t = \int_z^\infty \frac{dz}{H_0(1+z)\sqrt{\Omega_m(1+z)^3+\Omega_\Lambda}}. \quad (8)$$

The ratio T we used for the first step of the space partition is 50%. We only need to process the particle location information, so the data size to store one Gaussian component is 28 bytes. Based on the requirement and the condition, first of all, we select the number of Gaussian components through Equation 6. The data we used for this is the test dataset which comes from the GADGET-2 simulation using 2^{21} particles under the same simulation parameters.

By applying the information to the test dataset, we can calculate the total number of the GMMs when different Gaussian component number is given. Thus, *eAIC* of different Gaussian component numbers can be calculated when the compression rate is fixed. Increasing the number of components per GMM decreases the number of data proxy and increases particles for each model. The representation quality may not significantly improve when the particles and components increase at the same time. The result is shown in Figure 3 and the estimation took us about 5 minutes to calculate it on a single CPU. The line chart is the *eAIC* value in different numbers of Gaussian components. The histogram is the total number of GMMs at different numbers of Gaussian components. From the figure, we can see that the *eAIC* value is the smallest when the component number is 2. Therefore, we use the GMM with 2 Gaussian components in our application.

After the parameters were determined, we started the experiment. The experiment used 2^{30} particles and a cosmic space with a side length of 50 Mpc to simulate dark matter on 480 processors. The processor is Intel Xeon E5-2690. The simulation runs for 1,142,859.93 seconds about 317 hours. We saved 135 snapshots in total, covering the entire simulation. In the post-analysis, we use the reconstructed particle data to visualize and computer astronomical properties to prove the effectiveness of our method. Also, we calculated the same properties using the original data for comparison.

7.2 Performance Analysis

The data reduction method runs simultaneously to the simulation. When data storage is needed, the simulation program needs to stop and wait for the data reduction method. Our method will also increase memory usage because we need to create a k-d tree, an array of raw data indexes, the GMMs and related parameters in the leaf node. The k-d tree needs to keep the structure information of the tree, and the extra memory consumption is usually less than 1%. The length of the index array is the number of the particle and the extra memory consumption of this array is 12.5%. The total memory usage of the GMMs and related parameters is the size of the storage data, so the extra memory consumption is about 1%. Therefore, for each computing node, the additional memory usage is no more than 15% of the original.

The expected storage after in-situ data reduction can be calculated according to the compression ratio and the location information size of each snapshot. As the result the expected storage should be 164 megabytes for one snapshot. In the experiment, we store the raw data and our data proxy at the same time. We document the storage size of 135 records and show the results in Figure 4. The black line in the figure is the expected storage size and the red line is the actual storage size. From the results, we can see that I/O storage of our method has a steady performance. All the actual storage sizes are near the expected storage size. The fluctuation of storage size is caused by the number of particles on each processor being slightly different for each time step. We use about 21.6 gigabytes in total to save the data proxy and use 1620 gigabytes in total to save the position information of raw data. The final compression rate is as expected. Hence, our in-situ data reduction method meets the domain experts' requirement.

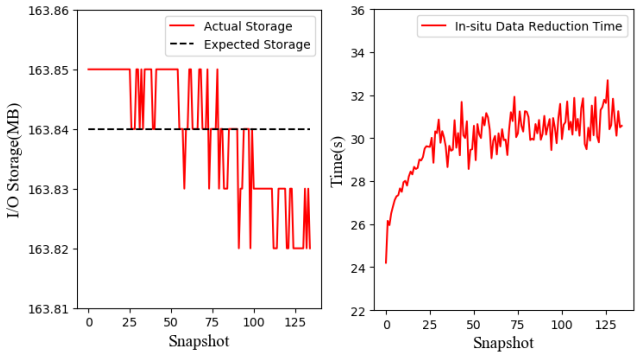


Figure 4: This figure shows the data storage size and the processing time for each snapshot. In the left figure, the black dashed line is the expected storage size and the red line is the actual storage size of our methods. In the right figure, the red line is the processing time of our in-situ data reduction algorithm. The total time of our method is about 0.35% of the total simulation time.

Besides, domain experts hope that our approach does not take up a long time for in-situ data processing, which includes the time for the simulation space partitioning, data modeling and data proxy I/O. We show the time of processing 135 snapshots in Figure 4. The in-situ data reduction time of each snapshot is the maximum running time of our algorithm on all computing nodes. In the figure, we can see that the running time tends to be stable as the simulation progresses. 135 records cost 4041.75 seconds and 0.35% of the total simulation time. Our domain experts are highly satisfied with the result of time occupation.

7.3 Quantitative Evaluation of Physical Property

The cosmological simulation focuses on the feature structures of the particle distributions rather than on a single particle. Therefore, it is more appropriate to use the actual physical properties to measure the error using our reduction method for cosmological data. In this section, we will calculate the error of the physical properties of the reconstructed data.

7.3.1 Matter Power Spectrum

To describe the statistical properties of large-scale matter distributions, astrophysicists usually adopt the matter power spectrum [12] which contains the information of the matter density field in the Fourier space. The exact definition of the matter power spectrum, $P(k)$, is

$$P(k) \equiv \left\langle |\tilde{\delta}(\boldsymbol{k})|^2 \right\rangle, \quad (9)$$

where $\tilde{\delta}(\boldsymbol{k})$ is the Fourier transform of the dimensionless density contrast $\delta(\boldsymbol{x}) \equiv (\rho(\boldsymbol{x}) - \bar{\rho})/\bar{\rho}$ (i.e., the normalized density difference at $\boldsymbol{x}$ with respect to the mean background density). The symbol $\langle \cdots \rangle$ means the ensemble average which averages over all $\boldsymbol{k}$ vectors with the same magnitude k.

The matter power spectrum is widely used in the analysis of cosmological N-body simulations and galaxy survey observations. It

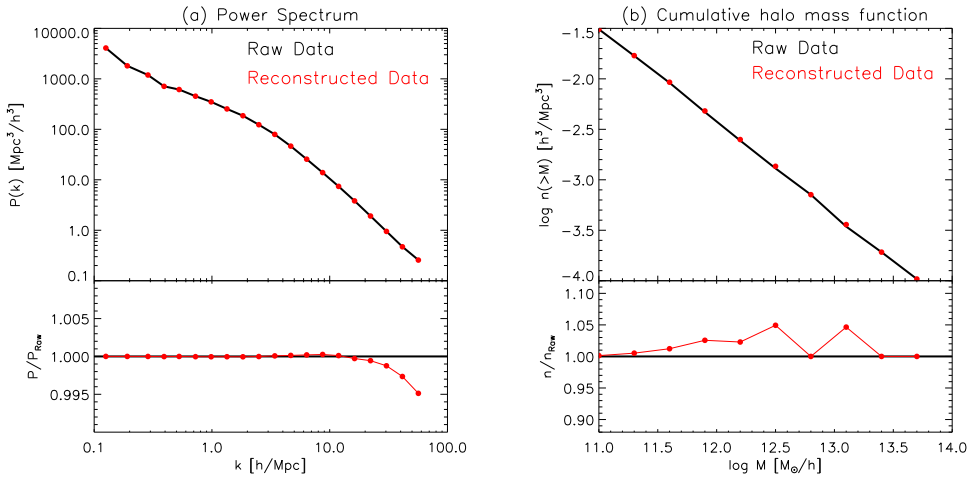


Figure 5: The above two diagrams show the power spectrum and cumulative halo mass function by the raw data and the reconstructed data. Left column: the upper panel shows the power spectra from the original raw snapshot (black) and the reconstructed data (red), with the x-axis being the wave vector k in the Fourier space and the y-axis being the value of power spectrum $P(k)$. In the lower panel, we plot the ratio between the power spectra computed from our reconstructed data and the raw data as a function of k. Right column: similar to the left column, but for halo mass functions. The difference between the reconstructed power spectrum (halo mass function) and the original one is less than 0.5% (4.9%).

encodes the information of cosmological parameters and the structure formation process, and thus it is a useful tool in constraining cosmological models.

We show the matter power spectra computed from the original simulation snapshot and our reconstructed data in the left panel of Figure 5, where the x-axis is the wavevector k in the Fourier space, the upper y-axis is the power spectrum $P(k)$, and the lower y-axis is the ratio between the power spectrum computed from the GMM reconstructed data and the one from the raw snapshot. The reconstructed data power spectrum only starts to deviate from the original power spectrum at small scales (i.e., $k > 20\ h\text{Mpc}^{-1}$). The difference between the reconstructed data and original power spectra is smaller than 0.5% up to $k = 60\ h\text{Mpc}^{-1}$, indicating that the in-situ data reduction method is able to recover the data which conforms to the power spectrum of a simulation fairly well.

7.3.2 Halo Mass Function

Another widely used tool in the analysis of simulations is the halo mass function [24], $n(>M)$ with n denoting the halo number density and M being the halo mass. The halo mass function tells us the abundance of dark matter halos distributing in the universe, i.e., the number density of dark matter halos with masses larger than M. The halo mass function also contains a wealth of cosmological information such as the model parameters and the nonlinear formation of structures.

The halo mass functions from the original simulation and reconstructed data are shown in the right panel of Figure 5. Here, x-axis is the halo mass M in logarithmic scale while the upper y-axis is the halo mass function $n(>M)$, and the lower y-axis is the ratio normalized to the original $n(>M)$. The average error between the reconstructed data and the original halo mass functions is about 1.6% in a wide range of halo mass, i.e., from $10^{11}\ h^{-1}M_{\odot}$ to $\sim 10^{14}\ h^{-1}M_{\odot}$. We can see that our in-situ data reduction method reproduces the halo abundance in a high precision, which meets the requirements in simulation data analysis.

7.4 Qualitative Evaluation in Visualization

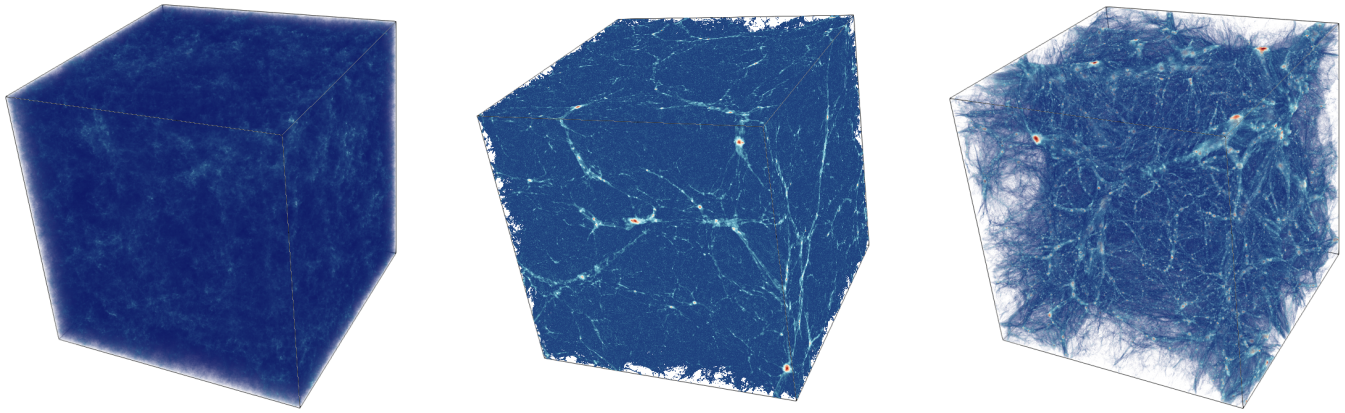
Visualization is one important way for domain experts to study their simulation output. Through the visualizations, the formation of the complex structure of the universe in the simulation can be clearly demonstrated. Domain experts can validate the theoretical model by studying the visualization results. Therefore, domain experts require that our data be used for visualization purposes, and can show the global evolution of the universe and the shape of specific cosmic structures. In this section, we compare the visualizations between the reconstructed data and the raw data.

7.4.1 The Evolution of The Universe

By visualizing the large-scale structure of the universe over time, domain experts can gain a better understanding of the evolution of the universe. After reconstructing the particle positions, we convert the data into volume data for visual analysis. In the process of data conversion, we first divide the three-dimensional space into cubic grid cells where the side length of the cell is $0.1Mpc$. After dividing, we count the number of particles in each cell and use the particle count as the voxel value. Then, we use the ParaView [2] to render the converted volume data.

In Figure 6, we show the beginning, middle and final stages of the universe simulation. In the early universe, particles were randomly distributed in the simulated space. At that time, no cosmic characteristic structure was formed. From (a) and (d) of Figure 6 we can see that the reconstructed data is almost identical to the raw data in the visualization results. With the evolution of simulation, the particles moved under gravity and began to gradually form the feature structure of the universe. In the middle stage of the universe simulation shown in Figures 6 (b) and (e), the particles began to form the halo, void and filament structures. By comparing the structures in the images, we can find that the feature structure of the reconstructed data in visual results is basically the same as that of raw data. In the later stage of the simulation, the structure of the cosmic feature is clear and stable. The latter stage of the universe simulation is shown in (c) and (f) of Figures 6.

By comparing the visualization results in the reconstructed data and raw data at different time steps, we can see that the visualization results of the two groups of data are almost identical. This means that our in-situ data reduction method can support visualization of the evolution process in universe simulation, which meets the domain expert requirements for visualization of the large-scale global



(a) Raw data at the early stages of cosmic evolution (redshift = 20)

(b) Raw data at the middle stages of cosmic evolution (redshift = 0.7)

(c) Raw data at the end stages of cosmic evolution (redshift = 0)

(d) Reconstructed data at the early stages of cosmic evolution (redshift = 20)

(e) Reconstructed data at the middle stages of cosmic evolution (redshift = 0.7)

(f) Reconstructed data at the end stages of cosmic evolution (redshift = 0)

Figure 6: Comparison of visualizations of the raw data and reconstructed data at different redshift.

evolution process.

7.4.2 The Features in High-resolution

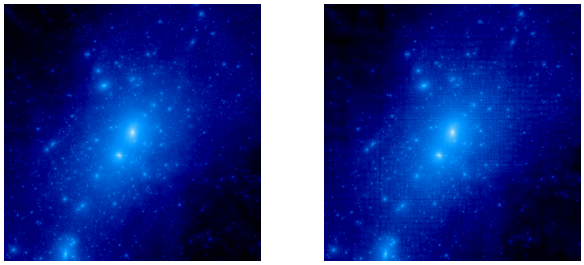


Figure 7: Zooming in on a halo. The left one is the raw data. The right one shows the reconstructed data.

In the study of cosmic structure, domain experts usually zoom in to a specific region and then analyze the local feature structures. Therefore, efficient local feature reconstruction and visualization are crucial for their data analysis needs. Because local feature reconstruction only needs partial data, reconstructing the whole data from our data proxy wastes a lot of time and computing resources. Our data proxy allows us to identify the leaf nodes that have an intersection with the zoomed area before reconstructing particles from the GMMs. Every leaf node represented by a GMM stored the mean particle location and a maximum distance to the mean particle location. This information defines a sphere that covers all particles of the leaf node. Our location feature reconstruction algorithm can go through all leaf nodes and only reconstruct particles in the leaf nodes which intersect with the zoomed area for visualization.

Another visualization result is also shown in Figure 7. The left sub-figure is the raw data, and the right sub-figure is the reconstructed data. We used images with 1024^2 pixels to show the spatial area with $5Mpc$ edge length. The figure shows a halo structure at the end of the simulation. Domain experts mainly focus on the shape of the halo and the general distribution of sub-halos(the brightened dot in the picture) of the halo. By comparison, we can see that the reconstructed data completely preserves the shape and structure of the halo. Domain experts commented that the visualization result is accurate enough to satisfy their analysis requirements.

7.4.3 Discussion

The visualization results of reconstructed data are very similar to the raw data, but there are still some differences. It is difficult to quantify differences from visual inspect only, so we calculate errors by comparing the voxel values. We normalize the value of voxels in volume data before calculating the errors in each snapshot. In Figure 8, we show the average errors of the voxel values. From the figure, we can see that the error is about 3% at the beginning of the

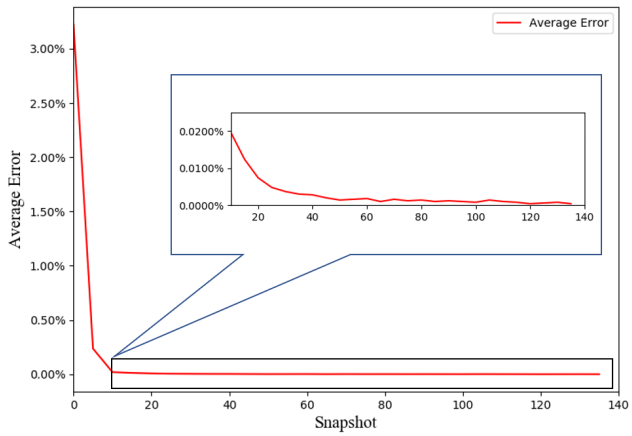


Figure 8: This chart shows the average error in volume data for the snapshots. The largest error in the figure is 3.22%. The least error in the figure is 0.0004%.

simulation and decreases rapidly with the evolution of simulation. We think that the value range of the voxels is small initially because the cosmic structure has not formed. As the simulation evolved, some cosmic structures began to emerge, which resulted in increased value ranges in the volumes.

In Figure 7, the result shows a halo in high-resolutions. The shape of the halo and the distribution of sub-halos can be clearly displayed. But some visible differences can be seen in the image. These patterns mostly occur around our leaf node boundaries because we use the Monte Carlo algorithm in data reconstruction. Random sampling can lead to those errors and display them in visualization. To analyze the error more accurately, we compare the value of pixels in the two images. We normalize those values before calculating the errors. The maximum error is 7.73%, the minimum error is 0%, and the mean error is 0.03%. The pixel point with the maximum error appears near the highest point of halo density because there are fewer reconstructed particles at the leaf node boundaries and this error will be enlarged in the high-density region. Our domain experts think that the image quality meets their needs because the structure and shape of the halo are complete, the mean error of the pixel values is small, and the error of the global physical properties is small. Therefore, these minor artifacts areas do not bother them when performing their data analysis tasks and visualization.

8 Error Study

In this section, we will discuss the errors of our algorithm under different compression rates and different particle numbers. We also compare our algorithm with the state-of-the-art compression methods. The following equation is used to calculate the error.

$$error = |1 - \frac{V_{rec}}{V_{raw}}| \tag{10}$$

V_{rec} is the value of physical property of the reconstructed data. V_{raw} is the value of physical property of raw data. We use this equation to calculate the error of the power spectrum and the halo mass function.

8.1 The Error Under Different Parameters

Under different parameters, the algorithm has different performance. We compare the errors for different numbers of particles at different compression rates. The number of astronomical particle data we used in this study was 2^{21}, 2^{24}, 2^{27} and 2^{30}. These datasets are from simulations with the same cosmological parameters. We compress the position information of particles in these datasets and the compression rate was 5%, 2%, 1%, and 0.5%. The comparison results are shown in Table 1 and Table 2.

Table 1: The average error in power spectrum under different compression rates and different particle numbers.

	Compression Rate			
Particle Number	5%	2%	1%	0.5%
2^{21}	1.15%	2.76%	4.55%	7.70%
2^{24}	0.25%	0.56%	1.20%	1.85%
2^{27}	0.07%	0.12%	0.36%	0.80%
2^{30}	0.03%	0.05%	0.05%	0.34%

Table 2: The average error in halo mass function under different compression rates and different particle numbers

	Compression Rate			
Particle Number	5%	2%	1%	0.5%
2^{21}	7.60%	14.35%	22.25%	28.29%
2^{24}	1.85%	2.66%	4.20%	8.50%
2^{27}	2.16%	1.05%	2.26%	3.43%
2^{30}	1.32%	2.20%	2.20%	3.24%

From the results, we can see that as the number of particles increased, the error of the reconstructed data decreased because the number of GMMs in each cosmic structure increased. Through this result, we can provide guidance to control error for domain experts. It can further help them to select the compression rate under different application requirements.

8.2 Comparisons of Different Compression Methods

At present, some studies have used compression algorithms to compress astronomical particle data. Zeyen et al. [36] compared the performance of various compression methods on astronomical particle data. Through his experimental results, we can see that the SZ compression algorithm has good performance. The SZ algorithm is a commonly used astronomical particle compression algorithm [8]. By adjusting the parameters, the compression rate of the SZ algorithm is close to 5%. Then we compare the errors between the SZ algorithm and our algorithm through the same dataset. The result of the errors in the power spectrum is shown in Table 3. From the table, we can see that the SZ compression algorithm has a large error at 5% compression rate. Therefore, our algorithm has better performance at a very high compression rate.

9 Summary

In this paper, we design an in-situ data reduction method for large-scale N-body cosmological simulations. In the in-situ processing stage, we use a combined k-dimensional partitioning and GMMs approach to reduce the data. We propose a method based on information theory to select the number of Gaussian components. After the processing, the size of data to be output is greatly reduced. In the post-analysis stage, we use the Monte Carlo algorithm to reconstruct the data. By calculating the power spectrum and halo mass function,

Table 3: The average error in the power spectrum using different compression method when the compression rate is 5%.

Particle Number	Error of Our Algorithm	Error of SZ Algorithm
2^{21}	1.15%	>100%
2^{24}	0.25%	>100%
2^{27}	0.07%	>100%
2^{30}	0.03%	>100%

we show that the error of the reconstructed data is small, which meets the requirements of the domain experts for data analysis. By comparing the visualization results of raw data and the reconstructed data, we show that our method can meet the requirements of experts for visualization analysis. Therefore, our in-situ data reduction method can support large-scale data analysis and visualization. Domain experts have thought highly of our methods.

In future work, we plan to visualize the trained GMMs directly without random sampling to save more computing resources.

Acknowledgments

This work was supported by the National Key R&D Program of China (2017YFB0203300) and the Informatization Plan of Chinese Academy of Sciences (XXH13503-07).

References

[1] J. Ahrens, S. Jourdain, P. OLeary, J. Patchett, D. H. Rogers, P. Fasel, A. Bauer, M. Petersen, F. Samsel, and B. Boeckel. In situ mpas-ocean image-based visualization. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, Visualization & Data Analytics Showcase*, 2014.

[2] U. Ayachit. *The paraview guide: a parallel visualization application*. Kitware, Inc., 2015.

[3] J. Bentley. Multidimensional binary search trees used for associative searching. communications of the acm, 18(9), 509-517. *Commun. ACM*, 18:509–517, 09 1975.

[4] B. Binotto, J. L. Comba, and C. Freitas. Real-time volume rendering of time-varying data using a fragment-shader compression approach. In *Proceedings of the 2003 IEEE Symposium on Parallel and Large-Data Visualization and Graphics*, page 10. IEEE Computer Society, 2003.

[5] C.-M. Chen, A. Biswas, and H.-W. Shen. Uncertainty modeling and error reduction for pathline computation in time-varying flow fields. In *2015 IEEE Pacific Visualization Symposium (PacificVis)*, pages 215–222. IEEE, 2015.

[6] P. Collaboration. Planck 2015 results. xiii. cosmological parameters. *Astronomy & Astrophysics*, 594:A13, 06 2016.

[7] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via em algorithm. *J. Royal Statistical Soc., Series B*, 39:1 – 38, 09 1977.

[8] S. Di and F. Cappello. Fast error-bounded lossy hpc data compression with sz. In *2016 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 730–739, 2016.

[9] S. Dutta, C.-M. Chen, G. Heinlein, H.-W. Shen, and J.-P. Chen. In situ distribution guided analysis and visualization of transonic jet engine simulations. *IEEE transactions on visualization and computer graphics*, 23(1):811–820, 2017.

[10] S. Dutta, H.-W. Shen, and J.-P. Chen. In situ prediction driven feature analysis in jet engine simulations. In *2018 IEEE Pacific Visualization Symposium (PacificVis)*, pages 66–75. IEEE, 2018.

[11] S. Dutta, J. Woodring, H.-W. Shen, J.-P. Chen, and J. Ahrens. Homogeneity guided probabilistic data summaries for analysis and visualization of large-scale data sets. In *2017 IEEE Pacific Visualization Symposium (PacificVis)*, pages 111–120. IEEE, April 2017.

[12] D. Eisenstein and W. Hu. Power spectra for cold dark matter and its variants. *The Astrophysical Journal*, 511, 10 1997.

[13] H. Guo, W. He, T. Peterka, H.-W. Shen, S. M. Collis, and J. J. Helmus. Finite-time lyapunov exponents and lagrangian coherent structures in uncertain unsteady flows. *IEEE transactions on visualization and computer graphics*, 22(6):1672–1682, 2016.

[14] S. Guthe and W. Strasser. Advanced techniques for high-quality multi-resolution volume rendering. *Computers & Graphics*, 28(1):51–58, 2004.

[15] W. He, C.-M. Chen, X. Liu, and H.-W. Shen. A bayesian approach for probabilistic streamline computation in uncertain flows. In *2016 IEEE Pacific Visualization Symposium (PacificVis)*, pages 214–218. IEEE, 2016.

[16] D. Jensen. Mixture models: Theory, geometry and applications. *Journal of Statistical Planning and Inference - J STATIST PLAN INFER*, 59:179–181, 03 1997.

[17] C. Li and H.-W. Shen. Winding angle assisted particle tracing in distribution-based vector field. In *SIGGRAPH Asia 2017 Symposium on Visualization*, page 16. ACM, 2017.

[18] S. Liu, J. A. Levine, P.-T. Bremer, and V. Pascucci. Gaussian mixture model based volume visualization. In *IEEE Symposium on Large Data Analysis and Visualization (LDAV)*, pages 73–77. IEEE, 2012.

[19] E. B. Lum, K.-L. Ma, and J. Clyne. Texture hardware assisted rendering of time-varying volume data. In *Proceedings Visualization, 2001. VIS'01.*, pages 263–563. IEEE, 2001.

[20] K.-L. Ma. In situ visualization at extreme scale: Challenges and opportunities. *IEEE Computer Graphics and Applications*, 29(6):14–19, 2009.

[21] K. Moreland, N. Fabian, P. Marion, and B. Geveci. Visualization on supercomputing platform level ii asc milestone (3537-1b) results from sandia. *Technical report SAND 2010-6118, Sandia National Laborato-ries, Tech. Rep.*, 2010.

[22] D. Posada and T. R. Buckley. Model selection and model averaging in phylogenetics: advantages of akaike information criterion and bayesian approaches over likelihood ratio tests. *Systematic biology*, 53(5):793–808, 2004.

[23] D. Potter, J. Stadel, and R. Teyssier. PKDGRAV3: beyond trillion particle cosmological simulations for the next era of galaxy surveys. *Computational Astrophysics and Cosmology*, 4:2, May 2017.

[24] W. Press and P. Schechter. Formation of galaxies and clusters of galaxies by self-similar gravitational condensation. *The Astrophysical Journal*, 187:425–438, 01 1974.

[25] M. Rosenblatt. Remarks on some non-parametric estimates of a density function. *The Annals of Mathematical Statistics*, 27, 09 1956.

[26] V. Springel. cosmological simulation code gadget. *Monthly Notices of The Royal Astronomical Society - MON NOTIC ROY ASTRON SOC*, 364:1105–1134, 12 2005.

[27] V. Springel, S. White, A. Jenkins, C. Frenk, N. Yoshida, L. Gao, J. Navarro, R. Thacker, D. Croton, J. Helly, J. Peacock, S. Cole, P. Thomas, H. Couchman, A. Evrard, J. Colberg, and F. Pearce. Simulations of the formation, evolution and clustering of galaxies and quasars. *Nature*, 435:629–636, 06 2005.

[28] A. Tikhonova, C. D. Correa, and K.-L. Ma. Visualization by proxy: A novel framework for deferred interaction with volume data. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1551–1559, 2010.

[29] V. Vishwanath, M. Hereld, and M. E. Papka. Toward simulation-time data analysis and i/o acceleration on leadership-class systems. In *2011 IEEE Symposium on Large Data Analysis and Visualization*, pages 9–14. IEEE, 2011.

[30] K.-C. Wang, N. Shareef, and H.-W. Shen. Image and distribution based volume rendering for large data sets. In *2018 IEEE Pacific Visualization Symposium (PacificVis)*, pages 26–35. IEEE, 2018.

[31] T.-H. Wei, S. Dutta, and H.-W. Shen. Information guided data sampling and recovery using bitmap indexing. In *2018 IEEE Pacific Visualization Symposium (PacificVis)*, pages 56–65. IEEE, 2018.

[32] J. Woodring, J. Ahrens, T. J. Tautges, T. Peterka, V. Vishwanath, and B. Geveci. On-demand unstructured mesh translation for reducing memory pressure during in situ analysis. In *Proceedings of the 8th International Workshop on Ultrascale Visualization*, page 3. ACM, 2013.

[33] J. Woodring, J. P. Ahrens, J. Figg, J. Wendelberger, S. Habib, and K. Heitmann. In-situ sampling of a large-scale particle simulation for interactive visualization and analysis. *ieee vgtc conference on visualization*, 30(3):1151–1160, 2011.

[34] J. Woodring, M. Petersen, A. Schmeiber, J. Patchett, J. Ahrens, and H. Hagen. In situ eddy analysis in a high-resolution ocean climate model. *IEEE transactions on visualization and computer graphics*, 22(1):857–866, 2016.

[35] H. Yu, C. Wang, R. W. Grout, J. H. Chen, and K.-L. Ma. In situ visualization for large-scale combustion simulations. *IEEE computer graphics and applications*, 30(3):45–57, 2010.

[36] M. Zeyen, J. P. Ahrens, H. Hagen, K. Heitmann, and S. Habib. Cosmological particle data compression in practice. In *Proceedings of the In Situ Infrastructures on Enabling Extreme-Scale Analysis and Visualization*, pages 12–16, 2017.