In-Situ Data Reduction for AMR-Based Cosmology Simulations

Daocé Wang  
Washington State University  
Pullman, WA, USA  
daocewang@wsu.edu

Jesus Pulido  
Los Alamos National Laboratory  
Los Alamos, NM, USA  
pulido@lanl.gov

Pascal Grosset  
Los Alamos National Laboratory  
Los Alamos, NM, USA  
pascalgrosset@lanl.gov

Sian Jin  
Washington State University  
Pullman, WA, USA  
sian.jin@wsu.edu

Jiannan Tian  
Washington State University  
Pullman, WA, USA  
jiannan.tian@wsu.edu

James Ahrens  
Los Alamos National Laboratory  
Los Alamos, NM, USA  
ahrens@lanl.gov

Dingwen Tao  
Washington State University  
Pullman, WA, USA  
dingwen.tao@wsu.edu

ABSTRACT

Large-scale cosmology simulations generate large amounts of data for post-analysis, resulting in I/O and storage bottlenecks. This work investigates an effective in-situ error-bounded lossy compression for Nyx, an adaptive mesh refinement (AMR) based cosmology application. Our contribution is threefold: (1) We explore the best-fit in-situ error-bounded lossy compressor (including SZ and TTHRESH) for Nyx considering both compression ratio and post-analysis quality. (2) We propose an approach to adaptively optimize the compressor for different AMR levels based on our developed metric and data characteristics. (3) Our evaluation shows that our approach can improve the compression ratio by 1.7× over the baseline (i.e., from 66 to 116) with the same post-analysis quality.

1 INTRODUCTION AND MOTIVATION

Adaptive mesh refinement and Nyx application. Adaptive mesh refinement (AMR) is a method of adapting the accuracy of a solution by using a non-uniform grid for increasing computational and storage savings. Nyx is an AMR-based code that investigates the evolution of the universe. The size of data generated by Nyx could be prodigious, which makes it urgent to introduce lossy compression to further save time and space for users. As shown in Figure 1, AMR-based Nyx uses different resolutions in different data regions.

Lossy compression for AMR data. Compared to lossless compression, error-bounded lossy compression can offer tens to hundreds of higher compression ratios while controlling the loss of post-analysis quality [2]. Some previous works [5, 7] have studied lossy compression for AMR data, but they still suffer from several significant drawbacks: (1) They do not target in-situ compression or consider application-specific post-analysis. (2) They do not optimize compression by utilizing data features (e.g., topology and sparsity) of different AMR levels, which can provide useful spatial information for more effective decorrelation (i.e., removing the data redundancy) in lossy compression. (3) They do not perform an evaluation on real-world HPC application datasets.

To this end, in this work, we explore the best-fit in-situ lossy compressor for AMR-based cosmology simulations and optimize it for each AMR level based on the data features with Nyx datasets.

2 METHODOLOGY AND EVALUATION

2.1 Best-fit Compressor

In this section, we explore the best-fit compressor in terms of power spectrum, an FFT-based post-analysis for the Universe’s matter distribution computed by a cosmology analysis tool called Gimlet.
Figure 3: Relative power spectrum error of block-wise-TTHRESH. Legend shows the number of subblocks and PSNR.

Figure 4: Work flow of Nyx and baseline compression

Because of the globality, TTHRESH seems more suitable than SZ for Nyx, but as our goal is in-situ compression, the speed is another major consideration. We test the CPU time of TTHRESH and SZ and find that TTHRESH is 100× slower than SZ (568 seconds vs 4.3 seconds). This is because the time complexity of HOSVD (the decoration step in TTHRESH) is very expensive (i.e., O(N^4) for N^3 input data), taking about 98% of the total compression time, which is infeasible for in-situ solution. Thus, we focus on SZ in this work.

2.2 Compressor Optimization

In this section, we propose an approach to optimize the compressor for AMR data based on the data characteristics of each AMR level.

The workflow of Nyx is shown in Figure 4, that is, Nyx will generate plot-file, which can be converted to 1d binary data. We then reshape the 1d data into 3d for post-analysis. A straightforward idea is to compress the 1d data for each AMR level directly. However, in that way, we will lose almost all the topology information, which is of vital importance for data compression as aforementioned.

Another approach is to compress the 3d data, but the distribution of the finest level is very complex and has many empty regions, as shown in Figure 5a. Therefore, we propose to use a sparse tensor representation (removing the empty regions) to optimize the topology: (1) Partition the data into subblocks. (2) Remove the empty blocks and place the remaining blocks into a 4D array. (3) Pass the 4D array to SZ for compression. (4) Emplace the subblocks in the decompressed 4D array back to the original data. Overall, we can both keep the topology information and reduce the data size.

An interesting problem with sparse tensor representation is how we should partition the data, or what the sub-block size should be. On the one hand, if it is too large, we have to fill the empty regions with many zeros, which would greatly reduce the compression ratio of SZ (called “zero-data boundaries”). On the other hand, a small subblock size results in too many boundaries between data (called “data-data boundaries”), which would also affect SZ. We find out that the subblock size depends on the sparsity of the data. For example, as shown in Figure 5, the finest level of z10 (an early timestep) is fairly spares, 76.8% of the data is empty, leading to a small block size. By contrast, the coarse level is much denser, only 23.2% of the data are empty, thus a large subblock size is needed.

Finally, we visualize the compression error with different sparsities and subblock sizes. For the sparse data, if we use a large subblock size, the zeros would significantly affect the quality of SZ, resulting in obvious error at the edge of each subblock, as shown in Figure 6c. For the dense data, if we use a small subblock size, the error would increase almost everywhere, as shown in Figure 6b, since we lose the topology information; if we use a large subblock size, the error is more acceptable, as shown in Figure 6d. After our optimization, SZ can significantly improve the compression over the baseline (i.e., from 66 to 116).

3 CONCLUSION AND FUTURE WORK

In this work, we find the best-fit in-situ lossy compressor for Nyx in terms of both compression ratio and post-analysis quality. We also optimize the compressor for each AMR level based on the data features. Our solution can deliver up to 116× of compression ratio with only 1% post-analysis quality loss. We plan to further optimize each AMR levels by using different compression configurations (e.g., error bound) and apply our approach to more AMR applications.

(a) Sparse data with small subblock (b) Dense data with small subblock (c) Sparse data with large subblock (d) Dense data with large subblock

Figure 6: Error slice for different sparsity and subblock sizes, they are all on the same scale, so more colorful means more error
REFERENCES


