16 Nov 2018

Improving Error-Bounded Compression for Cosmological Simulation

Sihuan Li

Sheng Di

Xin Liang
Missouri University of Science and Technology, xliang@mst.edu

Zizhong Chen

et. al. For a complete list of authors, see https://scholarsmine.mst.edu/comsci_facwork/1098

Recommended Citation

S. Li et al., "Improving Error-Bounded Compression for Cosmological Simulation," *Proceedings of the 30th ACM/IEEE International Conference for High Performance Computing, Networking, Storage, and Analysis (2018, Dallas, TX)*, Association for Computing Machinery (ACM), Nov 2018.

Improving Error-bounded Compression for Cosmological Simulation

Sihuan Li[1], Sheng Di[2], Xin Liang[1], Zizhong Chen[1], Franck Cappello[2,3]
[1]University of California, Riverside (USA); [2]Argonne National Laboratory (USA); [3]University of Illinois Urbana-Champaign (USA)

Abstract

Cosmological simulations may produce extremely large amount of data, such that its successful run depends on large storage capacity and huge I/O bandwidth, especially in the exascale computing scale. Effective error-bounded lossy compressors with both high compression ratios and low data distortion can significantly reduce the total data size while guaranteeing the data valid for post-analysis. In this poster, we propose a novel, efficient compression model for cosmological N-body simulation framework, by combining the advantages of both space-based compression and time-based compression. The evaluation with a well-known cosmological simulation code shows that our proposed solution can get much higher compression quality than other existing state-of-the-art compressors, with comparable compression/decompression rates.

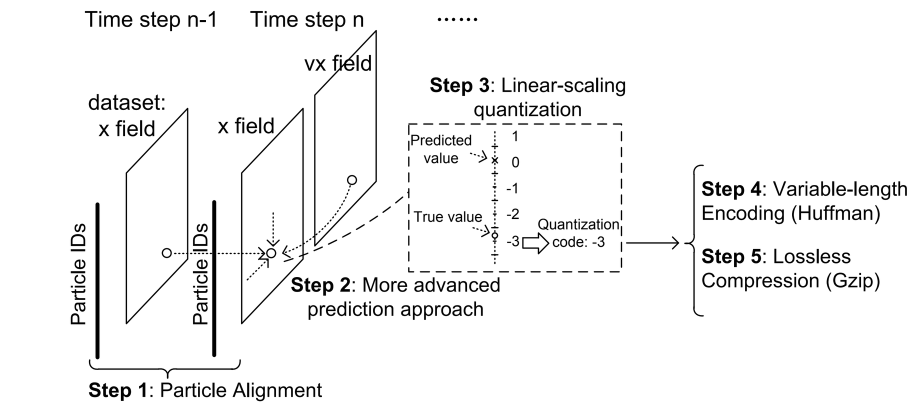
Introduction

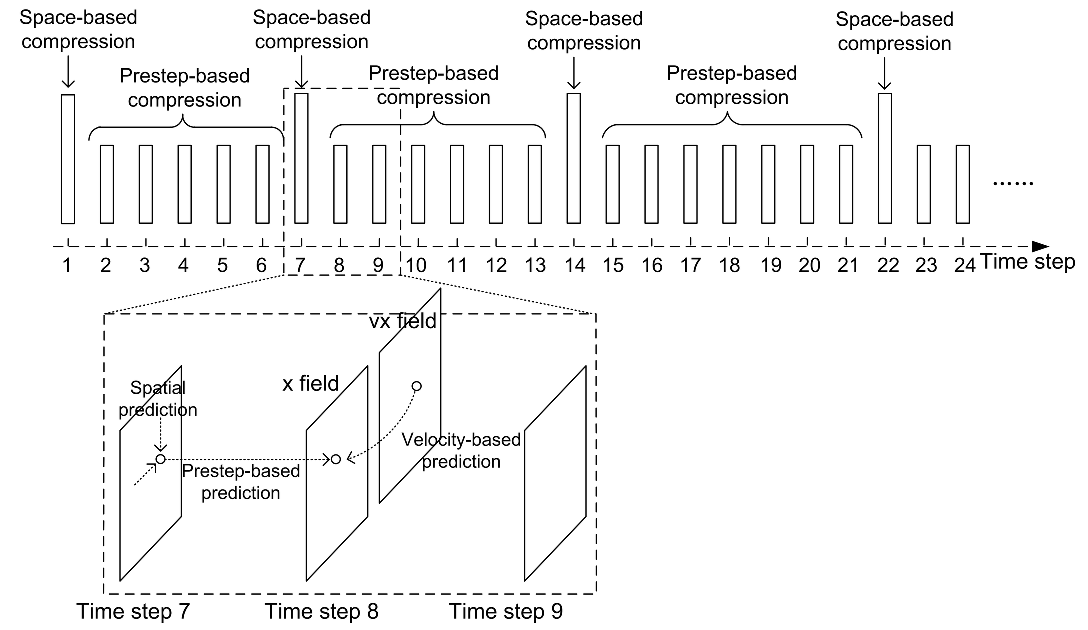
Scientific simulations of computing in the field of cosmology can produce extremely huge volume of data which needs to be stored in parallel file systems (PFS) in practice. Hardware/Hybrid Accelerated Cosmology Code (HACC) [5] is a typical example, which may produce 20+ PB of data even in one simulation case. How to reduce the cosmological N- body simulation data size significantly is fatal to the success of the cosmological simulations. Existing lossy compressors, however, mainly focus on the data consecutiveness in space, which is not suitable for the particle data because they are often stored based on separate dimensions (x, y, z) such that there are no spatial correlations for the adjacent data points.

In this work, we combine both space-based compression and time-based compression to improve the compression quality for cosmological data. Some challenges need to be resolved. First, the particles across consecutive time steps are not aligned generally, such that they are uneasy to compare across time steps. Moreover, the number of particles may be different between the snapshots at consecutive time steps. Secondly, how to combine space-based compression and time-based compression needs to be studied carefully. Last but not the least, we need to explore how to predict data values accurately with the consideration of strict error controls on demand. We make the following contributions:

- We propose a cosmological data compression model by combining both space and time dimension, with derived optimal intervals for the two types of compression.
- We propose an efficient on-line particle alignment making time-based compression possible for particle data
- Evaluation with a large-scale HACC simulation shows that our solution is the best in class.

Optimized Compression Framework for Cosmological Simulation





On-line Particle Alignment

The particle simulation data are not the same as the domain simulation data. They are not aligned between even between two continuous snapshots. Thus, to apply time based compression, we must align the data first. However, there exist two challenges in the simulation requirement: 1. We don't have more than 2 snapshot data in the memory at the same time; 2. The alignment has to be at most linear time to be comparable to compression which is also linear time. To handle this 2 challenges, we propose our efficient on-line particle alignment algorithm. The main idea to tackle the first challenge is: instead of aligning the particles actually, we use an array to indicate the alignment information to its previous snapshot. The main idea to tackle the second challenge is: before alignment, we do sorting that uses linear time and then find the overlapped particle. For the limitation of space, we do not show the detailed algorithm. We show an example in Figure 1 to demonstrate how the algorithm will work.

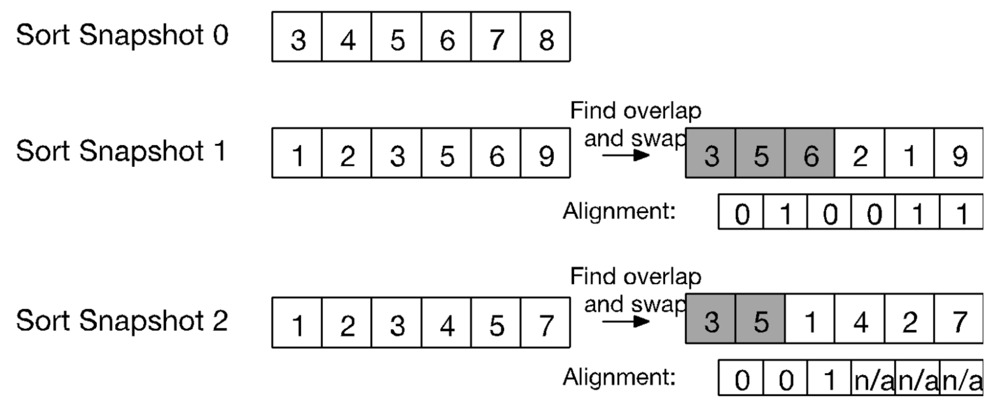


Figure 1 – Example work of our on-line particle alignment

The example shows only how we operate on the index of the data. In reality, the data needs to always obey the original alignment with their index. As Figure 1 shows, index of snapshot 1 needs to be aligned to index of snapshot 0 and index of snapshot 2 needs to be aligned to overlapped index of snapshot 1 and 0 (the dark zone of snapshot 1) and so on. The alignment array will store the information whether the previous snapshot index is existing in the current snapshot index ("0" for yes, "1" for no). So it should have the same length of the previous index array. Also, the overlapped index is alway the former part of the current snapshot index array (dark zone in Figure 1). With all these information, we can retrieve the corresponding data in previous snapshot to the current step. Thus time based compression is made possible for particle simulation.

Optimization of Space Based Compression Interval

A good question comes up in the time-space based compression framework. That is how the frequency of the snapshot based compression will be chosen or how many time-based compressions should operate between every snapshot based compression? We answer the question in the form of solving an optimization problem.

Assumptions and notations: Assume the particle overlap ratio with previous snapshot is α ($0 < \alpha < 1$);
Assume the reciprocal of space based compression ratio is a constant rs and that of time based compression ratio is a constant r_t ($0 < r_s, r_t < 1$). Without loss of generality, we set the total number of snapshots n, each snapshot has original file size 1, every space based compressed snapshot will be followed by $(k - 1)$ time based compressed snapshots.

Theorem 1: Given the above assumptions and notations, the optimal k to minimize the overall compressed file size is either $\lceil k_0 \rceil$ or $\lfloor k_0 \rfloor$ such that $\alpha^{k_0-1} (k_0 \ln\alpha - 1) + 1 = 0$.

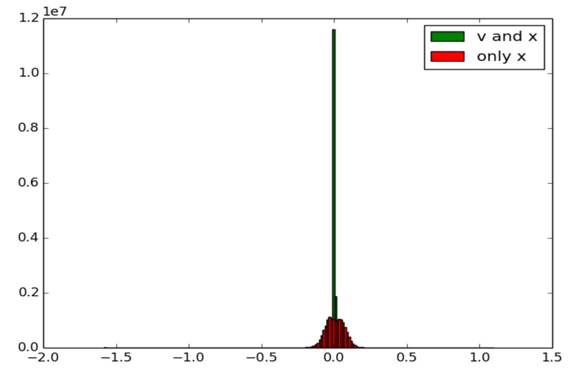


Figure 3 – Error distribution of our proposed prediction and traditional prediction on variable x

Considering the limited poster space, we will not give detailed proof for Theorem 1. However, the formalized the optimization problem is:

$$\begin{aligned} \min_{k} \quad & f(k) = n\left[r_s + \frac{\alpha}{1-\alpha}(r_t - r_s)\frac{1-\alpha^{k-1}}{k}\right] \\ \text{subject to} \quad & k \in [1, +\infty) \\ & k \in \mathbb{N} \end{aligned}$$

In the above formulations, f(k) is the overall compressed file size given the previous assumptions and notations. Thus, if we minimize the compressed file size, we can gain maximized overall compression ratio.
The detailed proof is omitted, however, the idea of the proof is that we show the objective function f(k) in the domain where $k > 1$ has only one global minimum. That means the root of the first order derivative function is the point where f(k) is minimized. That is just Theorem 1.

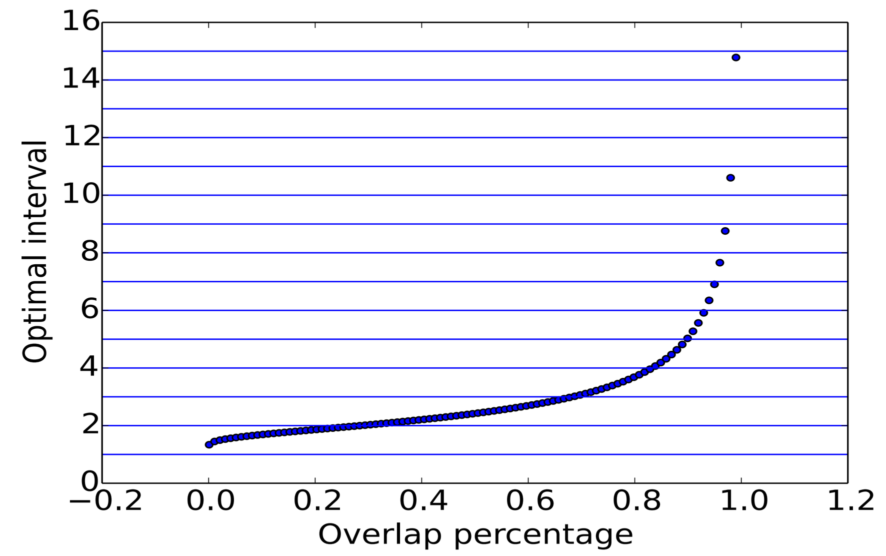


Figure 2 – Calculated optimal interval with increasing overlap percentage

The algorithm to find the optimal interval is very fast using binary search. Considering that we only need an integer solution instead of a real number solution, we can terminate the binary search as long as the search distance is less than 1. That means the complexity of the algorithm is $(\log i_{max})$ where i_{max} is the user defined maximum interval. Another good finding from our analysis is that the optimal interval only depends on the overlap percentage and has no dependence on other parameters like time based compression ratio (r_t) or space based compression ratio (r_s).

Optimization of Prediction Accuracy

Our compressor is based on data prediction. The more accurate the prediction is, the larger the compression ratio will be. To improve the prediction accuracy of position variable. We combine velocity and position information together in the previous snapshot to predict the position of the current step by leveraging Newton's law:
$x_1 = x_0 + v_0 \Delta t$. We minimize the mean square error (MSE) to get a good estimation of Δt, as formulated below:

$$\begin{aligned} \min_{\Delta t} \quad & mse(\Delta t) = \frac{1}{n}(\vec{x_1} - \vec{x_0} - \vec{v_0}\Delta t)^2 \\ \text{subject to} \quad & \Delta t \in (-\infty, +\infty) \\ & \Delta t \in \mathbb{R} \end{aligned}$$

This optimization problem can be solved uniquely in linear time. We show that the error distribution (in Figure 3) of the traditional prediction method which will only use the position of the previous step and the one we proposed. Intuitively, the sharper the error distribution is, the larger the compression ratio will be.

Evaluation

Users require value range based relative error for position and point-wise relative error for velocity. We evaluate the position variables and velocity variables separately. We do not compare with the TNG-MF1 trajectory compressor since number of particles is changing. In that case, TNG-MF1 compressor reduces to single-frame compression [6] We compress 100 snapshots of HACC data totaling 36GB. To evaluate position variables, we set the value range based error bound to be 1e-3, 1e-4, 1e-5, 1e-6 and compare our proposed compression method (SZ_vlct) with single snapshot based SZ (SZ_s) and time based SZ that only use previous step position information to predict the current one (SZ_prestep) and ZFP, Numarck, decimation. In SZ_vlct, to predict position, we fixed the point-wise relatively error bound to be 0.1 for velocity.

We plot the rate distortion for variable x in Figure 4. The results are similar for y and z. The figure demonstrates our proposed compression outperforms the other ones by a factor of more than 1.5X. Notice that Numarck and decimation are not able to compress particle data since they are not aligned. We test Numarck and decimation using reordered data that are generated beforehand offline.

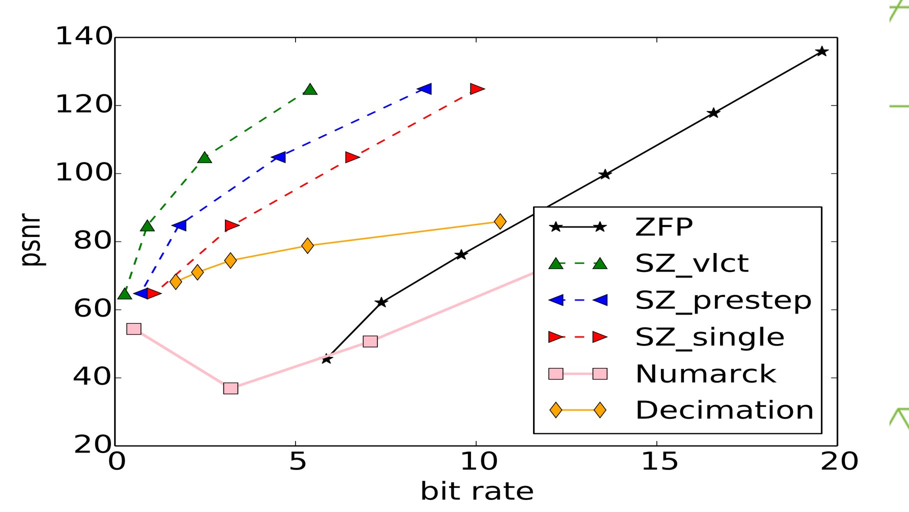


Figure 4 – Rate distortion for variable x of HACC data

To evaluate velocity variables, we set the point-wise relative error bound to be 1e-1, 1e-2, and 1e-3 and compare our method (SZ_vlct) with Numarck (Nk), single snapshot based SZ (SZ_s) and FPZIP. To evaluate compression rate, we fixed the value range based absolute error bound for position to be 1e-6 and the point-wise relative error bound for velocity to be 1e-3. Decimation and Numarck are omitted because of relatively low compression quality.

	1E-1				1E-2				1E-3			
	SZ_vlct	Nk	SZ_s	FPZIP	SZ_vlct	Nk	SZ_s	FPZIP	SZ_vlct	Nk	SZ_s	FPZIP
Vx	**13.90**	**11.28**	**9.33**	**5.67**	**7.59**	**5.69**	**4.91**	**3.71**	**4.64**	**2.97**	**3.24**	**2.75**
Vy	**12.96**	**10.78**	**8.66**	**5.33**	**7.22**	**5.49**	**4.69**	**3.56**	**4.45**	**2.92**	**3.14**	**2.66**
Vz	**11.60**	**10.08**	**7.42**	**4.98**	**6.69**	**5.21**	**4.24**	**3.40**	**4.19**	**2.91**	**2.93**	**2.58**

Table 1 – Compression ratio for velocity variables of HACC data

	SZ_vlct	SZ_s	ZFP + FPZIP
Compression rate	33.76	93.12	67.62
Decompression rate	98.80	64.96	65.09

Table 2 – Overall compression/decompression rate (MB/s)

For velocity, our compression ratios improve over Numarck by 15% to 56%. For compression rate, our methods slow down because of particle alignment but has better decompression rate.

Conclusion

We propose a general compression framework combining space-based compression and time-based compression. Several optimization techniques are proposed especially for cosmological scientific simulations based on the compression framework. Evaluation on one of the most popular large scale cosmological simulation (HACC) shows our methods outperform the state of the art.

References

[1] Tao, D., Di, S., Chen, Z. and Cappello, F., 2017, May. *Significantly improving lossy compression for scientific data sets based on multidimensional prediction and error-controlled quantization*. In Parallel and Distributed Processing Symposium (IPDPS), 2017 IEEE International (pp. 1129-1139). IEEE.
[2] Tao, D., Di, S., Chen, Z. and Cappello, F., 2017, December. *In-depth exploration of single-snapshot lossy compression techniques for N-body simulations.* In Big Data (Big Data), 2017 IEEE International Conference on (pp. 486-493). IEEE.
[3] Lindstrom, P., 2014. *Fixed-rate compressed floating-point arrays.* IEEE transactions on visualization and computer graphics, 20(12), pp.2674-2683.
[4] Yuan, Z., Hendrix, W., Son, S.W., Federrath, C., Agrawal, A., Liao, W.K. and Choudhary, A., 2016, December. *Parallel Implementation of Lossy Data Compression for Temporal Data Sets.* In High Performance Computing (HiPC), 2016 IEEE 23rd International Conference on (pp. 62-71). IEEE.
[5] S. Habib, A. Pope, H. Finkel, N. Frontiere, K. Heitmann, D. Daniel, P. Fasel, V. Morozov, G. Zagaris, T. Peterka, et al, "*Hacc: Simulating sky surveys on state-of-the-art supercomputing architectures*," In New Astronomy, 42:4965, 2016.
[6] Lundborg, M., Apostolov, R., Spångberg, D., Gärdenäs, A., van der Spoel, D., and Lindahl, E., 2014. *"An efficient and extensible format, library, and API for binary trajectory data from molecular simulations."* Journal of computational chemistry, 35(3), pp.260- 269.

The International Conference for High Performance Computing, Networking, Storage and Analysis(SC)'18 | Nov, 11-16, 2018, Dallas, TX, USA