Accelerating Relative-error Bounded Lossy Compression for HPC datasets with Precomputation-Based Mechanisms

Xiangyu Zou, Tao Lu, Wen Xia, Xuan Wang, Weizhe Zhang, Sheng Di, Dingwen Tao and Franck Cappello

Harbin Institute of Technology, Shenzhen & Peng Cheng Laboratory & Marvell Technology Group & Argonne National Laboratory & University of Alabama & University of Illinois at Urbana-Champaign

2019/5/23

Outline

- Background of research
- Our design
- Evaluation
- Conclusion

Background

- Scientific simulations
  - Climate scientists need to run large ensembles of high-fidelity 1kmX1km simulations. Estimating even one ensemble member per simulated day may generate **260** TB of data every **16s** across the ensemble.
  - Cosmologicaly simulation may produce **40PB** of data when simulating 1 trillion of particles in hundreds of snapshots.
- Data reduction is required
  - Lossless compression
    - Simulation data often exhibit high entropy
    - Reduction ratio usually around 2:1
  - Lossy compression
    - More aggressive data reduction scheme
    - High reduction ratio

Background - Lossy compressors

- ZFP
  - follow the classic texture compression for image data
  - Data transformation + embedded coding
  - **Low** compression ratio , **High** compression speed
- SZ
  - Prediction + quantization + Huffman encodng + Zstd
  - **High** compression ratio, **Low** compression speed
- A dilemma: which compressor should I use?

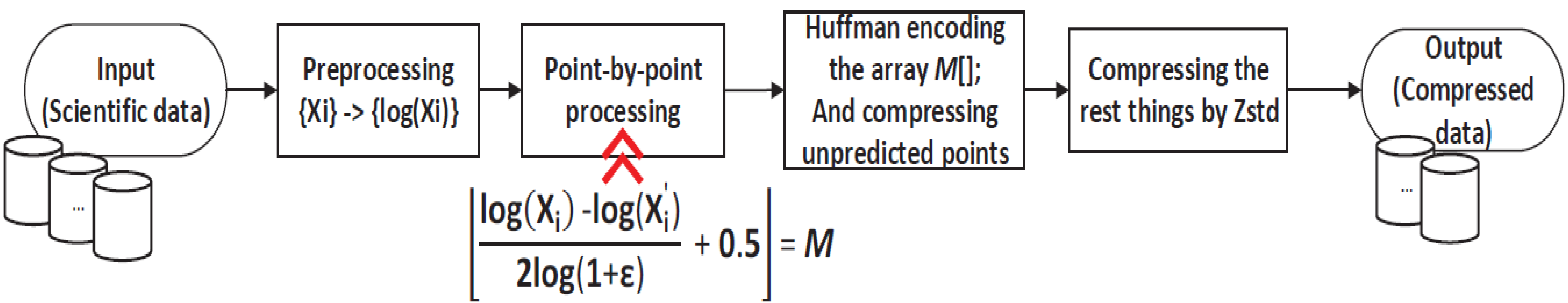
- Question: Can we significantly improve compression speed for SZ, leading to an easy solution for users?

Background - Lossy compression error bound

- Absolute error bound
  - For a value f, we get f' $\in$(f - ε, f + ε) is acceptable
- Pointwise relative error bound
  - For a value f, we get f' $\in$(f * (1 - ε), f * (1 + ε)) is acceptable
- CLUSTER18: Convert a pointwise relative error bound to an absolute error bound with a logarithmic transformation
  - log(f*(1 - ε))=log(f)+log(1 - ε), log(f*(1 + ε))=log(f)+log(1 + ε)
  - log(f') $\in$(log(f) + log(1 - ε), log(f) + log(1 + ε))

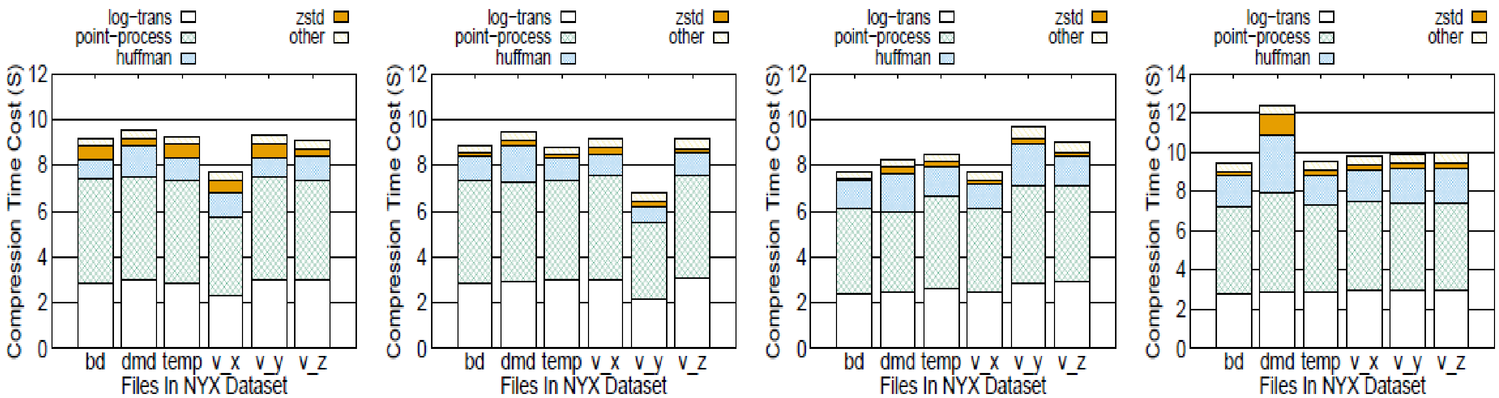
Background – design of SZ compressor for relative error control

- Preprocess - Logarithmic transformation
- Point-by-point processing – prediction & quantization
- Huffman encode
- Compression with lossless compressor

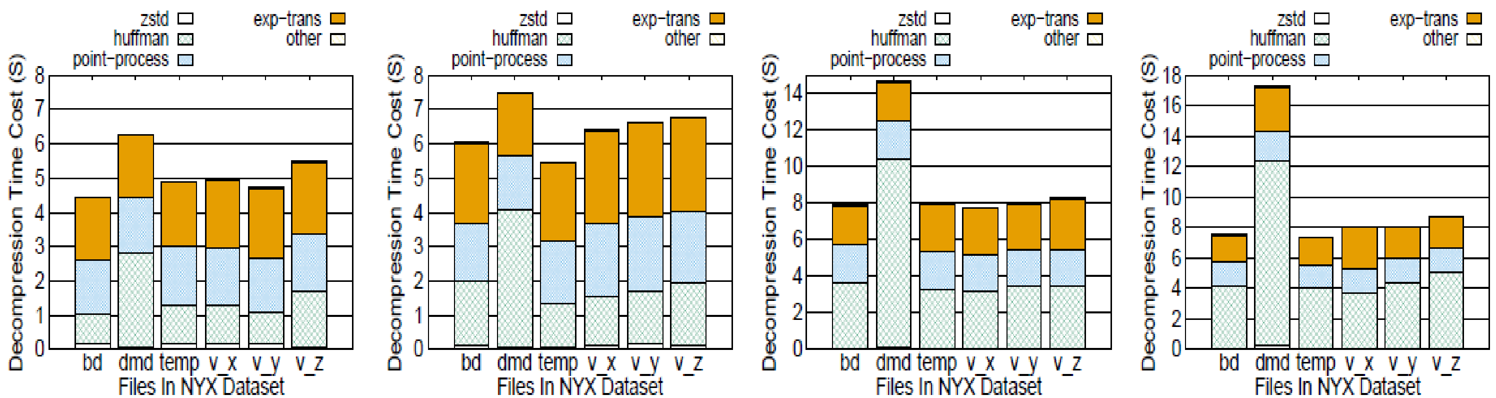


Logarithmic transformation (logX) is too expensive!

Performance breakdown of SZ Compression/Decompression



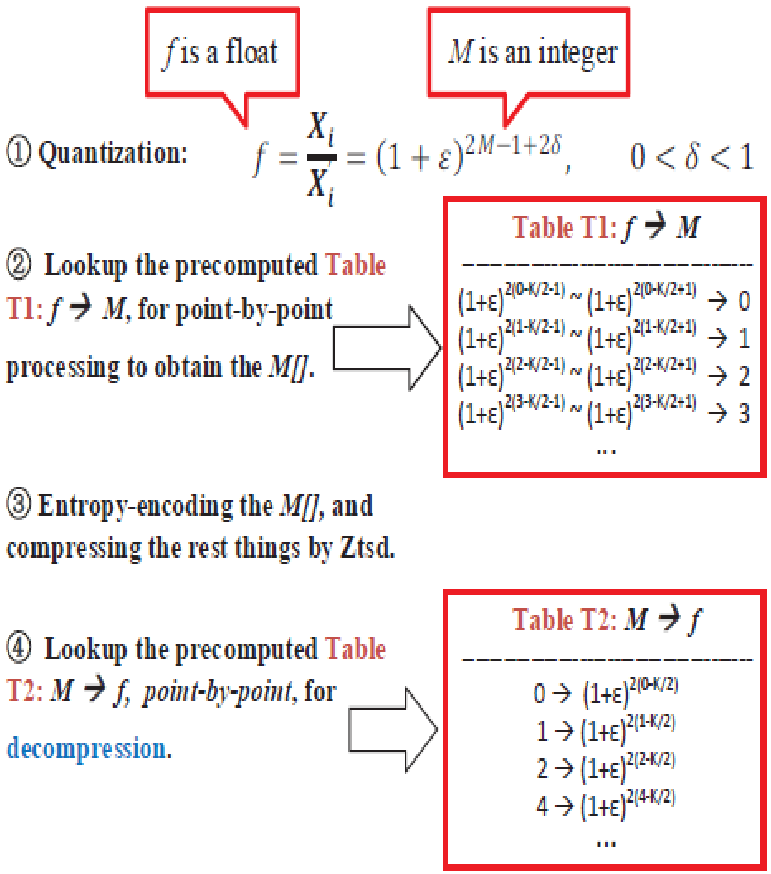
(a) 1.00E-01 (b) 1.00E-02 (c) 1.00E-03 (d) 1.00E-04



(a) 1.00E-01 (b) 1.00E-02 (c) 1.00E-03 (d) 1.00E-04

Time costs on log-trans and exp-trans stages consist about 1/3 of the total

Our design - workflow



- No longer to calculate the quantization factor, but look up tables.
- Using Table T1 to get quantization factor from f
- Using Table T2 to get a approximate value of f from quantization factor

Our design - Model A

- A simple idea
- We call ($(1+\varepsilon)^{2M-1}, (1+\varepsilon)^{2M+1}$) as PI(M)
- Separate the value field of f in to many grids.
- Each grid maps to a PI(M)

A general description to model A

$$f = \frac{X_i}{X_i'} \approx (1+\varepsilon)^{2M}$$

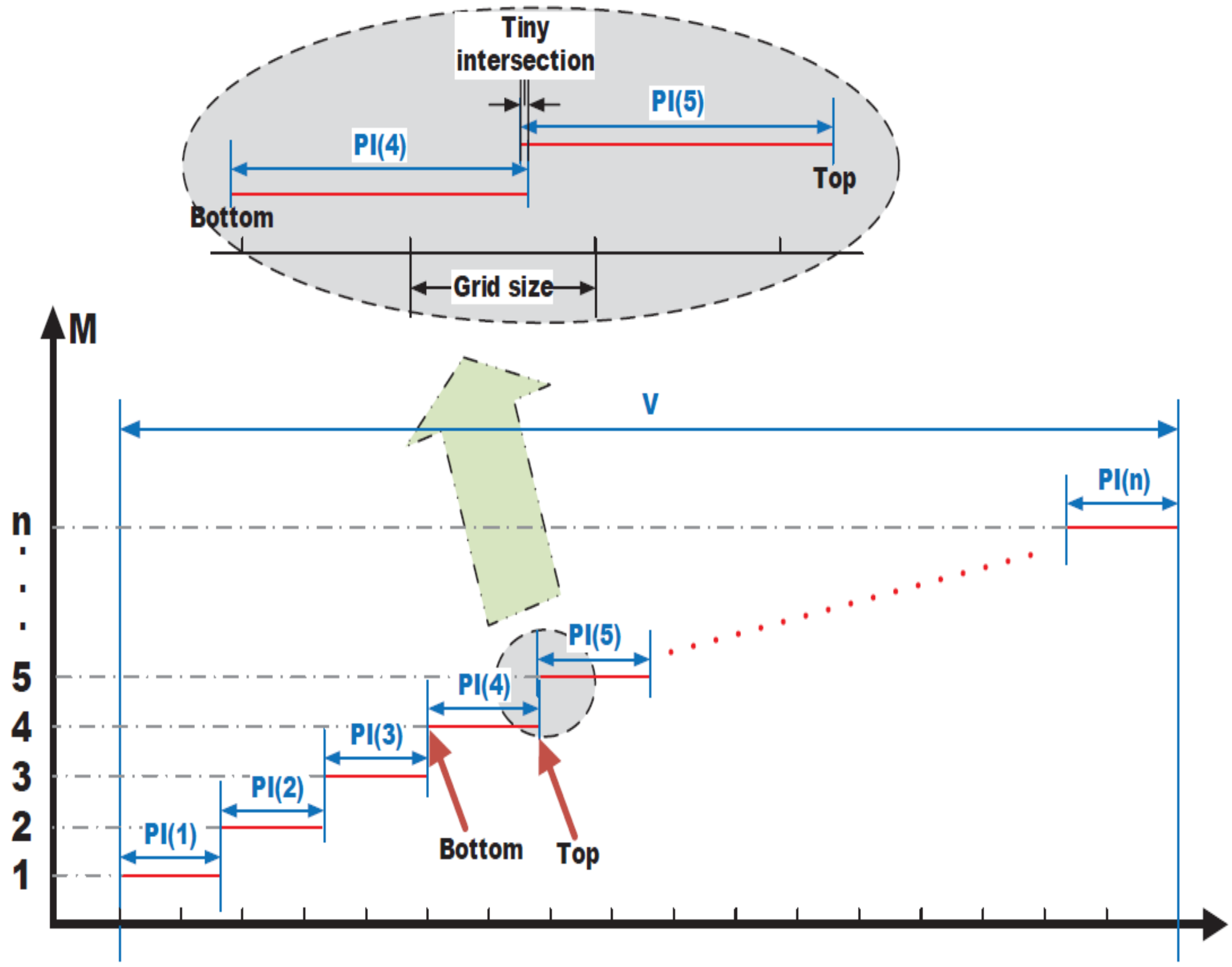
↓

$$(1+\varepsilon)^{2M-1} \le f \le \frac{(1+\varepsilon)^{2M}}{1-\varepsilon}$$

↓

PI interval

$$\left[(1+\varepsilon)^{(2M-1)}, \frac{(1+\varepsilon)^{2M}}{1-\varepsilon}\right]$$



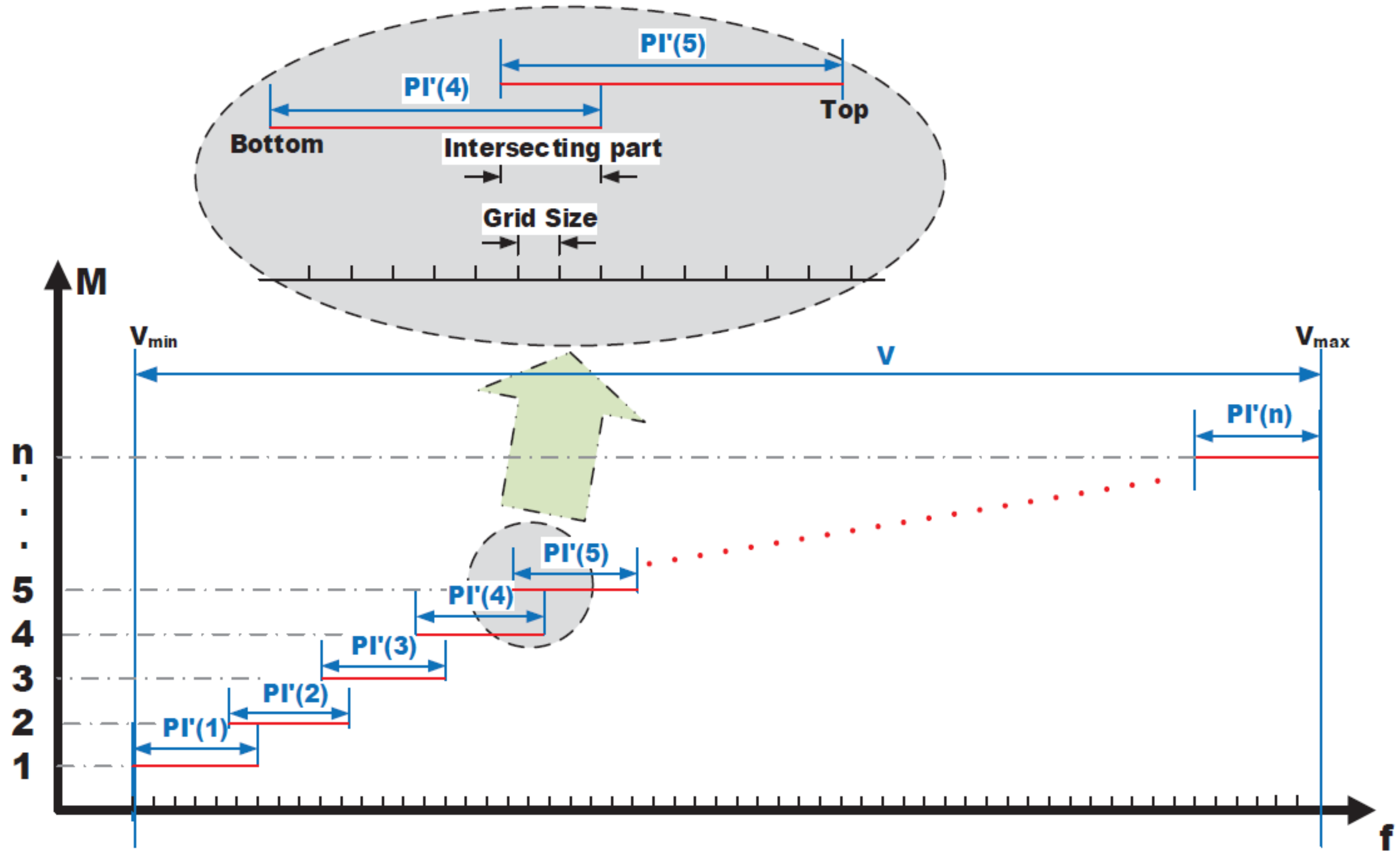
Our design - Model B

- Improved idea
- Let neighbor PIs have more intersection
- Let the size of grids is smaller than any intersection size
- We call ($(1+\varepsilon)^{M(2-\theta)-1}, (1+\varepsilon)^{M(2-\theta)+1}$) as PI'(M)
- Each grid maps to a PI'(M)

$$(1+\varepsilon)^{M(2-\theta)-1} \leq f' \leq \frac{(1+\varepsilon)^{M(2-\theta)}}{1-\varepsilon}, \quad 0 < \theta < 1$$

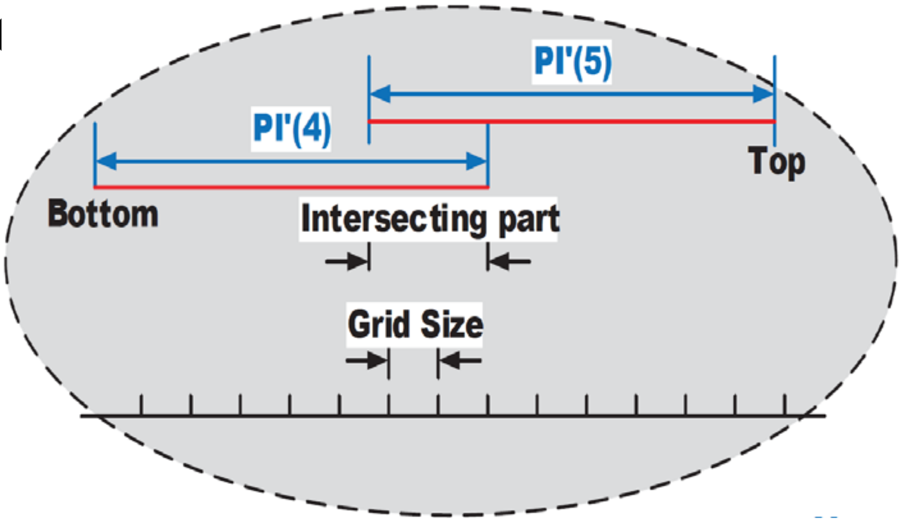
A general description about model P

Lemma 1. *If a grid size* **G** *is smaller than the size of any intersecting part of* $\boldsymbol{PI'}$, *a* $\boldsymbol{PI'}$ *completely including the grid always exists.*



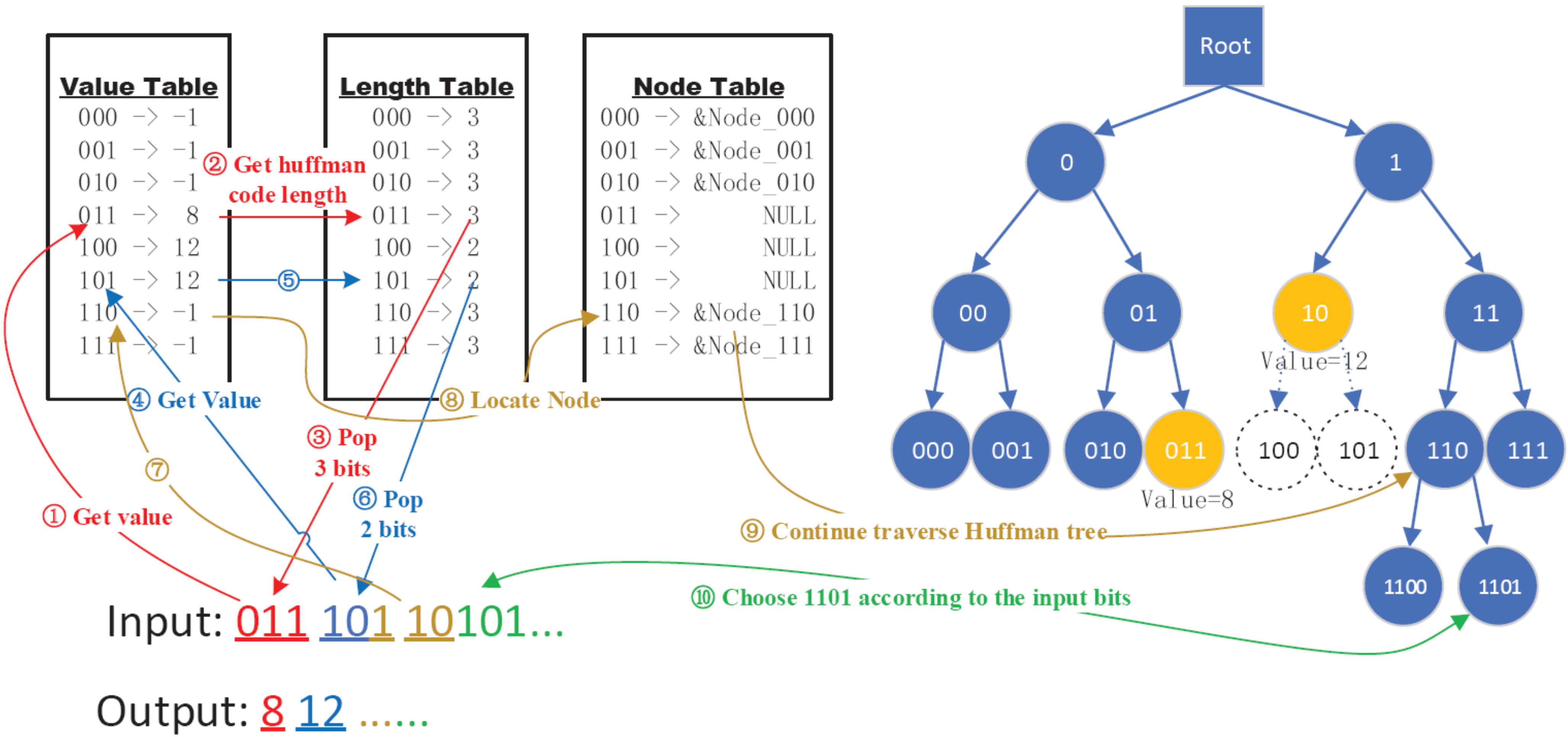
Our design - Advantage of Model B

- Any grid (i.e., a data point) is always included in a PI’
- Grid size is smaller than any intersection size, therefore any grid is complet PI’(M)
- Effect: Strictly respecting the bound



Accelerating Huffman decoding

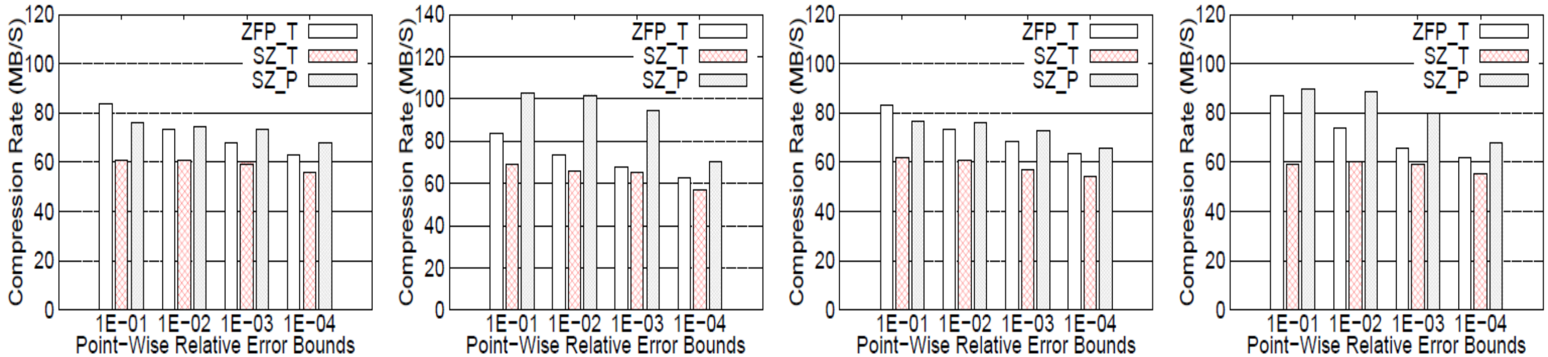
Idea: building precomputed table to accelerate Huffman decoding



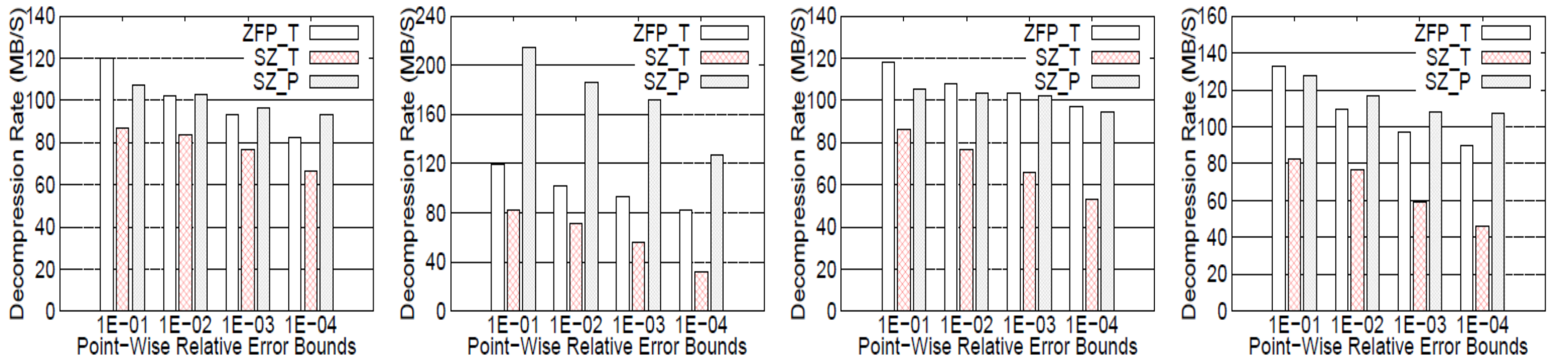
Performance Evaluation

- Environment
  - 2.4GHz Intel Xeon E5-2640 v4 processors
  - 256GB memory
- Datasets
  - NYX (3D, 3.1GB)
  - CESM (2D, 2.0GB)
  - Hurrican (3D, 1.9GB)
  - HACC (1D, 6.3GB)

Compression/Decompression Ra



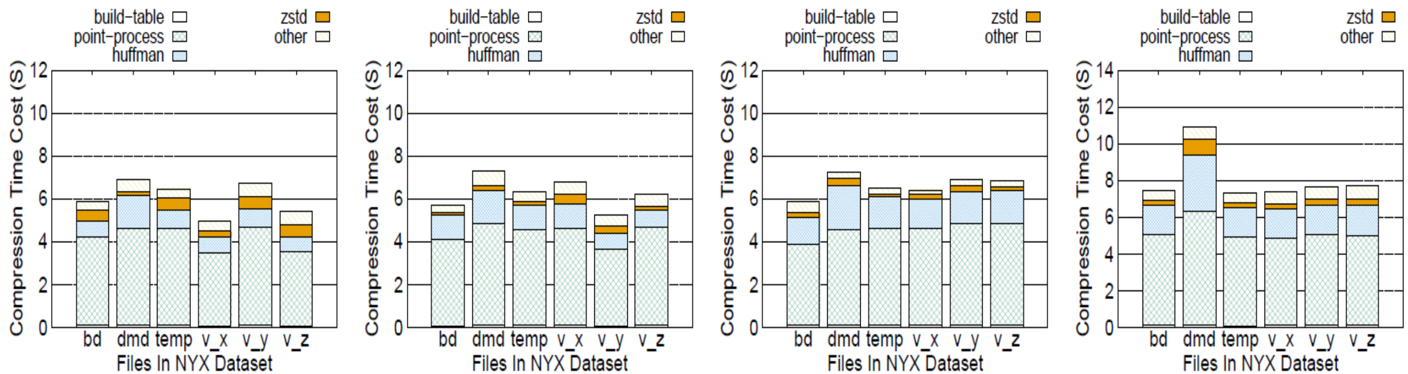
(a) CESM Dataset (b) HACC Dataset (c) Hurricane Dataset (d) NYX Dataset



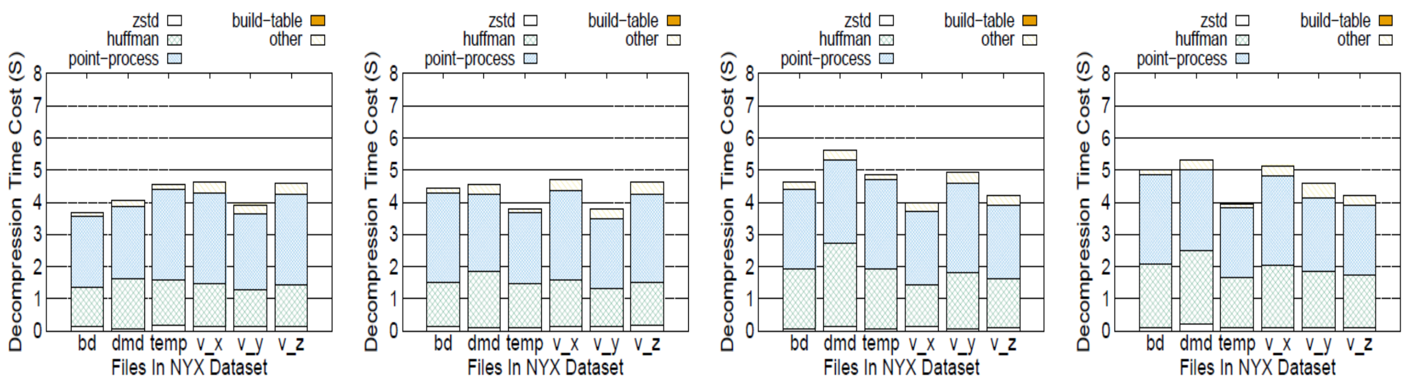
(a) CESM Dataset (b) HACC Dataset (c) Hurricane Dataset (d) NYX Dataset

Our Approach is about 1.2x ~ 1.5x than original SZ on compression rate and 1.3x ~ 3.0x on decompression rate。

Compression/Decompression breakdown



(a) 1.00E-01 (b) 1.00E-02 (c) 1.00E-03 (d) 1.00E-04

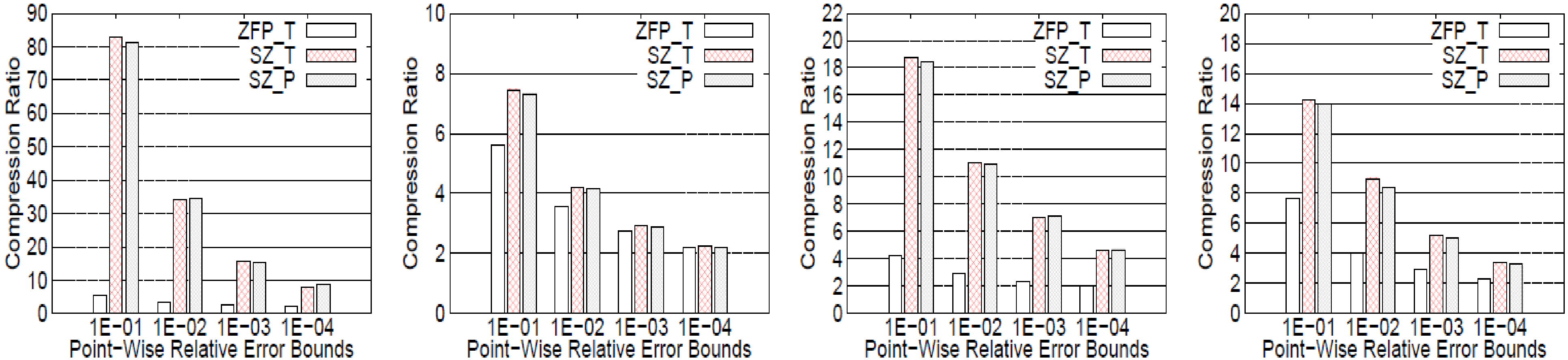


(a) 1.00E-01 (b) 1.00E-02 (c) 1.00E-03 (d) 1.00E-04

No time cost on log-trans and exp-trans. Time cost on build-table stage is very small.

Compression Ratio

We can observe that our solution (SZ_T) has very similar compression ratios with SZ_T.



(a) CESM Dataset (b) HACC Dataset (c) Hurricane Dataset (d) NYX Dataset

Data quality

POINT-WISE RELATIVE ERROR BOUND ON 3 REPRESENTATIVE FIELDS IN NYX.

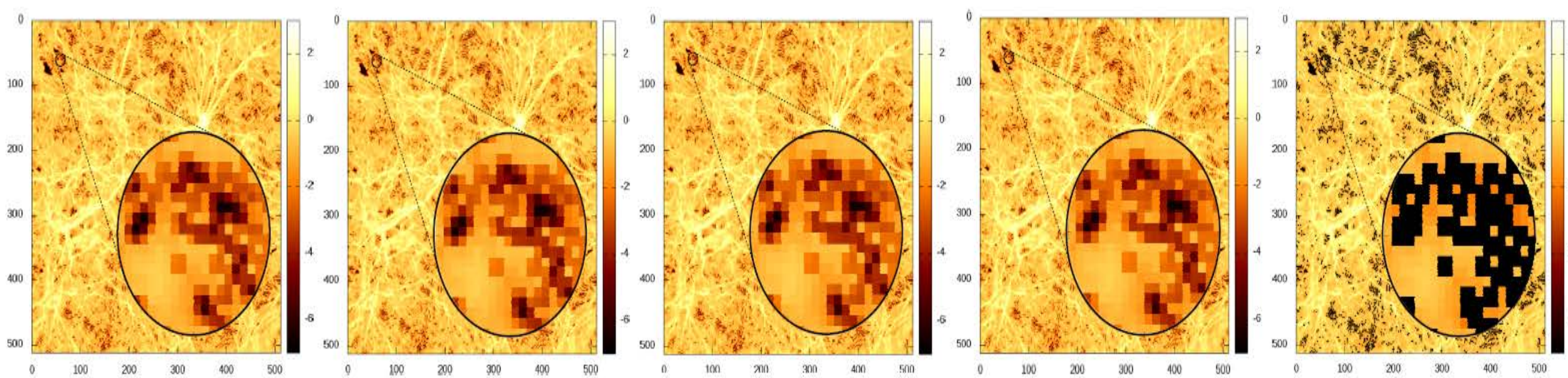
pwr_eb	type	dark_matter_density				velocity_x				temperature			
		MAX E	NRMSE	PSNR	CR	MAX E	NRMSE	PSNR	CR	MAX E	NRMSE	PSNR	CR
1E-01	SZ_P′	1.52E-01	3.97E-05	88.02	6.25	1.50E-01	2.27E-04	72.88	23.48	1.52E-01	4.70E-03	46.56	18.83
	SZ_T	1.00E-01	3.49E-05	89.13	6.19	1.00E-01	2.05E-04	73.75	24.97	1.00E-01	4.32E-03	47.29	19.85
	SZ_P	9.99E-02	3.26E-05	89.73	6.03	9.97E-02	1.96E-04	74.15	25.99	9.97E-02	4.10E-03	47.75	20.79
	ZFP_T	5.07E-02	4.64E-06	106.66	3.32	4.80E-02	2.89E-04	70.79	18.40	5.17E-02	2.74E-05	91.25	14.00
1E-02	SZ_P′	1.75E-02	4.05E-06	107.84	3.85	1.70E-02	2.36E-05	92.55	13.46	1.70E-02	5.39E-04	65.37	14.37
	SZ_T	1.00E-02	3.55E-06	108.99	3.85	1.00E-02	2.00E-05	93.97	14.06	1.00E-02	4.50E-04	66.93	13.55
	SZ_P	1.00E-02	3.42E-06	109.31	3.80	9.96E-03	1.82E-05	94.78	12.98	9.96E-03	4.23E-04	67.47	11.93
	ZFP_T	3.02E-03	2.75E-07	131.22	2.35	3.33E-03	3.45E-05	89.24	6.59	3.16E-03	1.75E-06	115.15	5.21
1E-03	SZ_P′	1.96E-03	4.30E-07	127.34	2.75	1.95E-03	2.58E-06	111.76	6.75	1.95E-03	6.70E-05	83.48	8.02
	SZ_T	9.97E-04	3.51E-07	129.10	2.74	9.98E-04	1.98E-06	114.08	6.61	9.98E-04	4.51E-05	86.91	7.63
	SZ_P	1.00E-03	3.44E-07	129.27	2.72	9.99E-04	1.79E-06	114.93	6.49	9.99E-04	4.25E-05	87.44	7.12
	ZFP_T	3.90E-04	3.56E-08	148.98	1.92	3.95E-04	4.63E-06	106.69	4.08	3.97E-04	2.23E-07	133.04	3.50
1E-04	SZ_P′	1.60E-04	3.96E-08	148.04	2.12	1.60E-04	2.10E-07	133.55	3.93	1.60E-04	4.98E-06	106.05	4.39
	SZ_T	9.80E-05	3.48E-08	149.16	2.09	9.90E-05	1.98E-07	134.05	3.92	9.90E-05	4.43E-06	107.08	4.38
	SZ_P	1.00E-04	3.38E-08	149.43	2.01	1.00E-04	1.72E-07	135.29	3.88	1.00E-04	4.25E-06	107.43	4.27
	ZFP_T	5.08E-05	4.49E-09	166.96	1.63	4.99E-05	5.81E-07	124.71	2.95	5.33E-05	2.76E-08	151.19	2.63

Comparable compression ratios with related works (SZ_T and ZFP_T)

Data quality (Cont'd)

Visualization of decompressed dark matter density dataset (slice 200) at the compression ratio of 2.75.

SZ series has a better visual quality than ZFP does.
SZ_P (both mode A and B) lead to satisfied visual quality!



(a) Original (b) SZ_P′ (Model A) (c) SZ_P (Model B) (d) SZ_T (e) ZFP_T

Conclusion

- We accelerate the SZ compressor for point-wise relative error bound control by designing a table-lookup method.
- We control the error bound strictly by an in-depth analysis of mapping relation between predicted value and quantization factor.
- Experiments show that 1.2x ~ 1.5x on compression speed and 1.3x ~ 3.0x on decompression speed, compared with SZ 2.1.

Thank you

Contact: Sheng Di (sdi1@anl.gov)