cuSZ: A High-Performance **GPU** Based **Lossy Compression** Framework for **Scientific Data**

Jiannan Tian	*Washington State University*
Sheng Di	*Argonne National Laboratory*
Kai Zhao	*University of California, Riverside*
Cody Rivera	*The University of Alabama*
Megan Hickman Fulp	*Clemson University*
Robert Underwood	*Clemson University*
Sian Jin	*Washington State University*
Xin Liang	*Oak Ridge National Laboratory*
Jon Calhoun	*Clemson University*
Dingwen Tao	*Washington State University*
Franck Cappello	*Argonne National Laboratory*

October 5, 2020 PACT '20, Virtual Event

Washington State University · UC Riverside · Clemson University · Argonne National Laboratory

Trend of Supercomputing Systems Gap Between Compute and I/O

The compute capability is ever growing while storage capacity and bandwidth are developing **more slowly** and not matching the pace.

supercomputer	year	class	PF	MS	SB	MS/SB	PF/SB
Cray Jaguar	2008	1 PFLOPS	1.75 PFLOPS	360 TB	240 GB/S	1.5k	7.3k
Cray Blue Waters	2012	10 PFLOPS	13.3 PFLOPS	1.5 PB	1.1 TB/S	1.3k	13k
Cray CORI	2017	10 PFLOPS	30 PFLOPS	1.4 PB	1.7 TB/S•	0.8k	17k
IBM Summit	2018	100 PFLOPS	200 PFLOPS	>10 PB••	2.5 TB/S	>4k	80k

PF: peak FLOPS MS: memory size SB: storage bandwidth

• when using burst buffer •• counting only DDR4

Source: F. Cappello (ANL)

Table 1: Three classes of supercomputers showing their performance, **MS** and **SB**.

Current Status of Scientific Applications: Big Data

application	data scale	passive solution (?)	to reduce
HACC cosmology simulation	**20 PB** per one-trillion-particle simulation	**use up FS** 26 PB for Mira@ANL	**10x** in need
CESM climate simulation	20% vs **50%** of h/w budget for storage 2013 vs **2017**	**5h30m** to store NSF Blue Waters, I/O at 1 TBps	**10x** in need
APS-U High-Energy X-Ray Beams Experiments	hundreds of **PB** brain initiatives	**100-PB** buffer or, connection at 100 GBps	**100x** in need

Error-Bounded Lossy Compression Matters

2:1 (FP-type)
lossless-compress scientific datasets

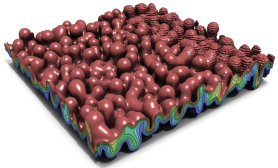
10:1 or higher
reduction ratio in need

industry lossy compressor
e.g., JPEG, MPEG

despite high reduction rate, **not** suitable for **HPC**
distinct in design goals

need **diverse** compression modes

1. absolute error bound (L^∞ norm)
2. pointwise relative error bound
3. RMSE error bound (L^2 norm)
4. fixed bitrate



Lossy compression for scientific data at varying reduction rate (10:1 to 250:1, left to right)
Figure from *Peter Lindstrom (LLNL)*

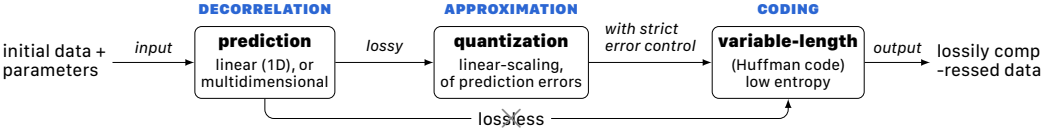
SZ [Di and Cappello 2016; Tao et al. 2017; Xin et al.2018]

- ▶ prediction-based lossy compressor framework for scientific data
- ▶ strictly control the global upper bound of compression error

SZ

`github.com/szcompressor/SZ`

SZ Framework (Error-Bound Workflow)



Motivation, Challenge, and Contribution

Research Objective and Contribution

- **cuSZ** IS THE FIRST STRICTLY ERROR-BOUNDED LOSSY COMPRESSOR ON GPU FOR SCIENTIFIC DATA.

Challenge

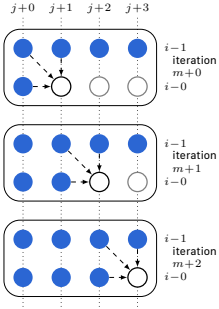
- Tight data dependency (loop-carried RAW) hinders parallelization.

  solution Eliminate dependency and parallelize it.

  DUAL-QUANTIZATION: {**PRE**, **POST**}QUANTIZATION

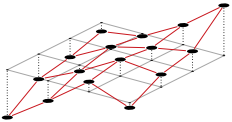
- Host-device communications only considering CPU/GPU suitableness.

  solution All tasks are done on GPU.
  Histograming [Gómez-Luna et al.]
  Customized Huffman codec (corse-grained)

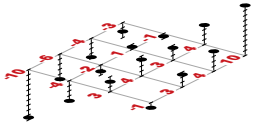


System Workflow Diagram of cuSZ

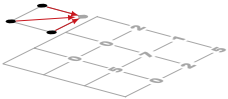
Dual-Quantization and Prediction



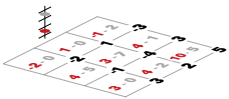
floating-point representation
original data



in units of *eb*
PREQUANTIZATION (NO RAW)



on PREQUANTIZATION set
ℓ-prediction results in unit weight
prediction (NO RAW)

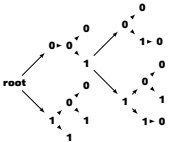


in units of *eb* (unchanged)
POSTQUANTIZATION (NO RAW)

Customized Huffman Encoding

```
                    range          freq.
|---------------+ 442—— 512        76%
|-----+           512—— 582        24%
|+                582—— 652      0.14%
|+                652—— 722     0.073%
|+                722—— 793     0.026%
|+                793—— 863    0.0095%
|+                863—— 933    0.0021%
|+                933——1024    0.00014%
```

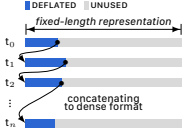
histograming



build and canonize Huffman codebook


```
quant.code      bitwidth ... Huff-code
    508         00000110 ... 00001010
    509         00000101 ... 00000100
    510         00000011 ... 00000100
    511         00000010 ... 00000001
    512         00000010 ... 00000011
    513         00000011 ... 00000101
    514         00000011 ... 00000000
    515         00000110 ... 00001100
```

memcpy fixed-length Huffman code



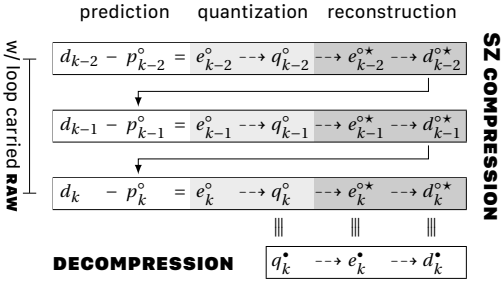
deflating Huffman codes

Loop-Carried Read-After-Write (P+Q) Procedure in SZ

- Lossless compression and decompression (codec) are mutually reversed procedures.
- Simlarly, SZ makes to-be-decompressed (reconstructed) data show during compression and make it under error control.
- Error control is conducted during quantization and reconstruction:

$$e^\circ/(2 \cdot eb) \times (2 \cdot eb) - e^\circ \leq eb.$$

- This introduces loop-carried read-after-write dependency.

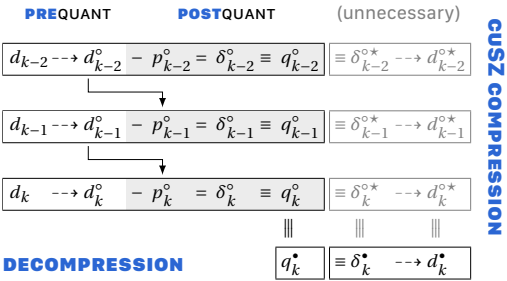


Fully Parallelized (P+Q) Procedure in cuSZ

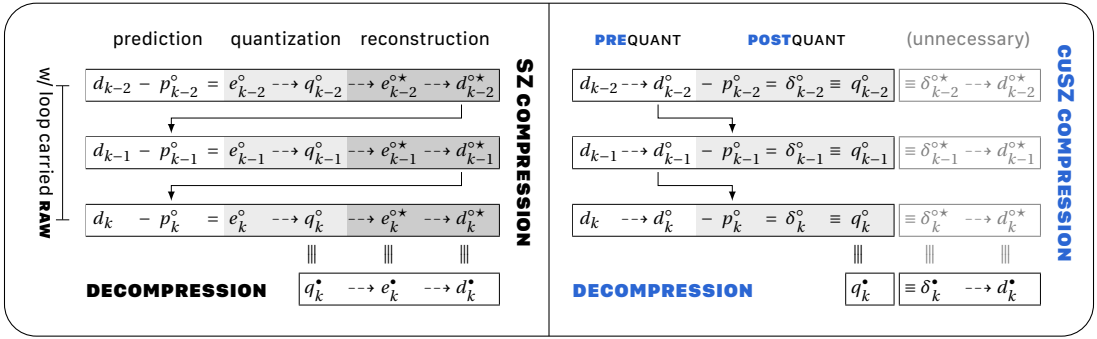
- Prioritize error control.
- Error control happens at the very beginning, **pre**quantization:

$$d^\circ/(2 \cdot eb) \times (2 \cdot eb) - d^\circ \leq eb,$$

- And **post**quantization is corresponding to quantization in SZ.



Original SZ (Loop-Carried RAW) vs Fully Parallelized cuSZ



Canonical Codebook and Huffman Encoding

ca·non·i·cal *adj.*

A canonical encoding is then generated in which the numerical values of the codes are monotone increasing and each code has the smallest possible numerical value consistent with the requirement that the code is not the prefix of any other code.

[Schwartz and Kallick 1964]

- codebook transformed to a compact manner
- no tree in decoding
- tree build time: 4-7 ms (for now)
- canonize for 200 us (1024 symbols)

- Encoding/decoding is done in a coarse-grained manner.
- A GPU thread is assigned to a data chunk.
- Tune degree of parallelism to keep every thread busy.

Mixture of Different Parallelisms

	sequential	coarse-grained	fine-grained	atomic
compression				
DUAL-QUANTIZATION			●	
histogram			●	●
build Huffman tree	●			
canonize codebook	●		●	●
Huffman encode (fix-length)			●	
deflate (fix- to variable-length)		●		
decompression				
inflate (Huffman decode)		●		
reversed DUAL-QUANTIZATION		●		

Table 2: Parallelism used for cuSZ's subprocedures (kernels) in compression and decompression.

Worth noting: in canonizing codebook

- problem size > max. `block` size (1024)
- utilize `cooperative groups` and `grid.sync()`
- `__syncthreads()`: not able
- `cudaDeviceSynchronize()`: expensive

Tuning Coarse-Grained Huffman Codec (Degree of Parallism/Concurrent Thread Numb

	HACC			CESM			HURRICANE			NYX			QMCPACK		
CHUNK	1071.8 MB	280,953,867	f32	24.7 MB	6,480,000	f32	95.4 MB	25,000,000	f32	512 MB	134,217,728	f32	601.5 MB	157,684,320	f32
SIZE	#THREAD	DEFLATE	INFLATE	#THREAD	DEFLATE	INFLATE	#THREAD	DEFLATE	INFLATE	#THREAD	DEFLATE	INFLATE	#THREAD	DEFLATE	INFLATE
2^{6}	.	.	.	1.0e5	11.3	25.0	.	.	.	.	.	.	.	.	.
2^{7}	.	.	.	5.1e4	15.5	37.8	.	.	.	.	.	.	.	.	.
2^{8}	.	.	.	**2.5e4**	**67.1**	**41.6**	9.8e4	5.1	11.0	.	.	.	.	.	.
2^{9}	.	.	.	1.3e4	55.6	30.7	4.9e4	10.2	9.4	.	.	.	.	.	.
2^{10}	.	.	.	6.3e3	48.2	19.6	**2.4e4**	**64.6**	**34.2**	1.3e5	4.7	5.9	1.5e5	4.7	5.1
2^{11}	1.4e5	4.6	2.8	.	.	.	1.2e4	57.3	27.7	6.6e4	5.7	6.3	7.7e4	5.2	6.2
2^{12}	6.9e4	5.1	5.1	.	.	.	6.1e3	50.7	17.8	3.3e4	25.1	16.1	3.8e4	12.9	11.1
2^{13}	3.4e4	13.6	12.1	.	.	.	.	.	.	**1.6e4**	**69.7**	**52.4**	**1.9e4**	**72.7**	**40.3**
2^{14}	**1.7e4**	**63.1**	**35.0**	.	.	.	.	.	.	8.2e3	72.4	42.6	9.6e3	75.9	29.0
2^{15}	8.6e3	65.8	28.1	.	.	.	.	.	.	4.1e3	50.0	23.1	4.8e3	56.0	16.1
2^{16}	4.3e3	45.9	14.3	.	.	.	.	.	.	.	.	.	.	.	.

Table 3: Throughputs (in GB/s) versus different numbers of threads launched on V100. The optimal thread number in terms of inflating and deflating throughput is shown in bold.

Evaluation Setup: Platform and Dataset

- Evaluation Platform (UA PantaRhei cluster)
  - NVIDIA **V100** (SXM2, 16 GB)
  - Dual 20-core Intel Xeon Gold 6148 CPUs
  - 16-lane PCIe 3.0 interconnect
- Comparison Baselines
  - (algorithmic) SZ-1.4.13.5: 16-bit quantization
  - (performance) cuZFP 0.5.5
- Test Datasets (from SDRB)
  - 1D **HACC**, cosmology particle simulation
  - 2D **CESM-ATM**, climate simulation
  - 3D **HURRICANE**, ISABEL simulation
  - 3D **NYX**, cosmology simulation
  - 4D **QMCPACK**, quantum Monte Carlo simulation

DATASETS	TYPE	DATUM SIZE / DIMENSIONS	#FIELDS / EXAMPLE(S)
COSMOLOGY HACC	fp32	1,071.75 MB / 280,953,867	6 in total / x, vx
CLIMATE CESM-ATM	fp32	24.72 MB / 1,800×3,600	79 in total / CLDHGH, CLDLOW
CLIMATE Hurricane	fp32	95.37 MB / 100×500×500	20 in total / CLOUDf48, Uf48
COSMOLOGY Nyx	fp32	512.00 MB / 512×512×512	6 in total / baryon_density
QUANTUM QMCPACK	fp32	601.52 MB / 288×115×69×69	2 formats in total / einspline

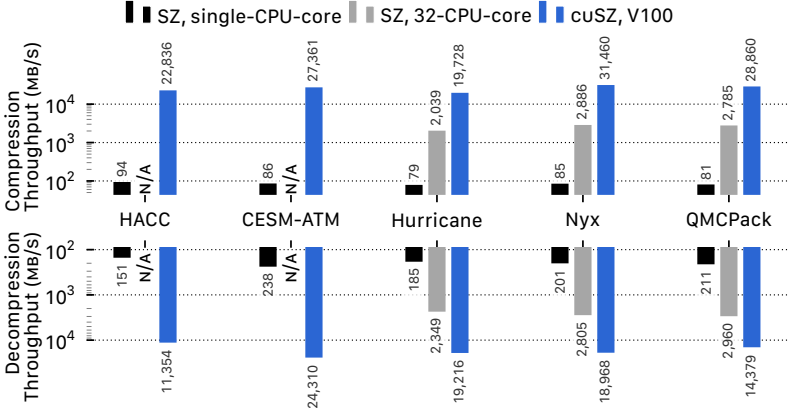
Table 4: Real-world datasets used in evaluation.

Breakdown Evaluation of Performance

		(P)REDICT. + (Q)UANT.	HUFFMAN CODING		KERNEL COMP.	GPU2CPU VALREL@10^{-4}	OVERALL COMPRESS	HUFFMAN DECODING	REVERSED (P+Q)	KERNEL DECOMP.
		MB/s	MB/s				MB/s	MB/s	MB/s	MB/s
CPU-SZ	HACC	137.7	328.6		-	-	94.1	196.0	659.3	151.1
	CESM-ATM	105.0	459.1		-	-	85.5	502.2	451.9	237.9
	HURRICANE	93.8	504.0		-	-	78.5	524.5	306.8	185.0
	NYX	98.5	648.7		-	-	84.7	670.4	300.5	201.8
	QMCPACK	97.5	396.2		-	-	80.8	660.3	313.4	211.1
		GB/s	HISTO. GB/s	DICT. MS	ENC. GB/s	GB/s	GB/s	CANONICAL DEC. GB/s	GB/s	GB/s
cuSZ	HACC	207.7	602.8	5.16	54.1	40.0	53.2	22.8	35.0	16.8	11.4
	CESM-ATM	252.1	345.3	4.33	57.2	41.1	81.9	27.4	41.6	58.5	24.3
	HURRICANE	175.8	418.0	4.81	55.2	38.2	40.8	19.7	34.2	43.9	19.2
	NYX	200.2	427.6	3.84	58.8	41.1	134.1	31.6	52.4	29.7	19.0
	QMCPACK	189.6	346.1	4.09	61.0	40.7	99.2	28.9	40.3	22.4	14.4
cuZFP	HACC	-	-	-	-	-	-	-	-	-	-
	CESM-ATM	-	-	-	-	47.6	27.7	17.5	-	-	113.1
	HURRICANE	-	-	-	-	83.7	27.7	20.8	-	-	102.2
	NYX	-	-	-	-	71.3	56.3	31.7	-	-	103.1
	QMCPACK	-	-	-	-	72.6	42.5	26.8	-	-	115.5

Table 5: Breakdown comparison of kernel performance among CPU-SZ, cuSZ, and cuZFP. "-" for N/A.

Performance Evaluation: Serial, OpenMP and CUDA



For compression,

242.9× to 370.1×

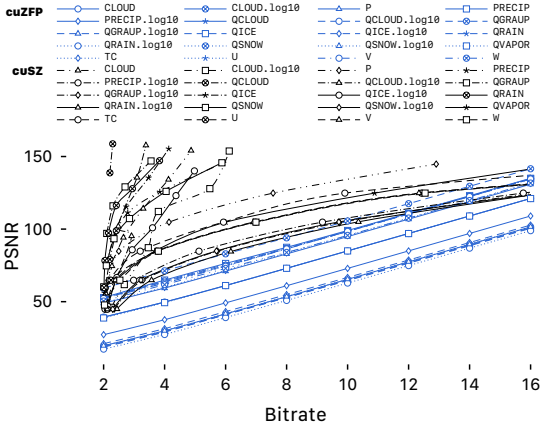
the serial CPU throughput

11.0× to 13.1×

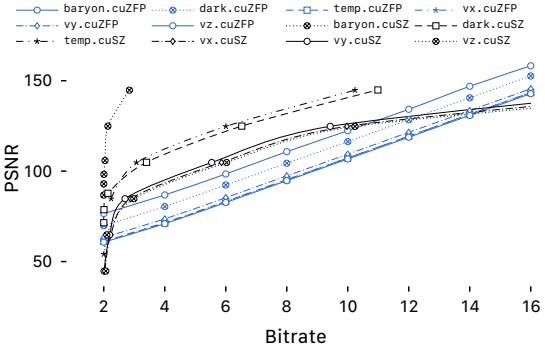
the OMP CPU throughput

Decompression alike.

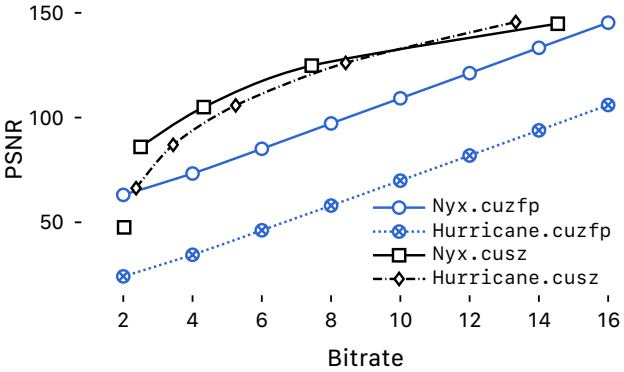
Evaluation of Compression Quality in Rate-Distortion (1/2)



85 dB (left: **HURRICANE**, right: **NYX**)



Evaluation of Compression Quality in Rate-Distortion (2/2)



Acknowledgement (Exascale Computing Project)

This R&D was supported by the Exascale Computing Project (ECP), Project Number: 17-SC-20-SC, a collaborative effort of two DOE organizations – the Office of Science and the National Nuclear Security Administration, responsible for the planning and preparation of a capable exascale ecosystem. This repository was based upon work supported by the U.S. Department of Energy, Office of Science, under contract DE-AC02-06CH11357, and also supported by the National Science Foundation under Grants SHF-1617488, SHF-1619253, OAC-2003709, OAC-1948447/2034169, and OAC-2003624.

THANK YOU !!

ANY QUESTION? 🤔

github.com/szcompressor/cuSZ

EXASCALE COMPUTING PROJECT

Argonne NATIONAL LABORATORY

WASHINGTON STATE UNIVERSITY

UC RIVERSIDE

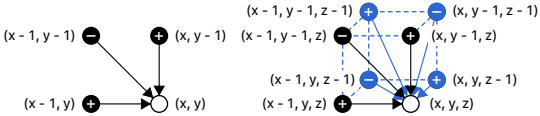
CLEMSON UNIVERSITY

BackUp (ℓ-Predictor)

- Gaussian-like, with signum altering to Manhattan distance to the (polarized) current point (■).

$$\mathsf{G}_{5\times5} = \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix} \quad \ell_{5\times5} = \begin{bmatrix} -1 & 4 & -6 & 4 & -1 \\ 4 & -16 & 24 & -16 & 4 \\ -6 & 24 & -36 & 24 & -6 \\ 4 & -16 & 24 & -16 & 4 \\ -1 & 4 & -6 & 4 & \blacksquare \end{bmatrix}$$

- Works for arbitrary dimension: from line to cube, to hypercube…



BackUp
(GPU Building Huffman Tree)

Sequential GPU building Huffman tree is too slow (1024 symbols for 4 ms).

- Improvement introduced by regularize memory access
  - Our preliminary improvement by switching to `thrust` is 4×, from 4 ms to 1 ms.
  - It is worthy maintaining workload on GPU for simplifying workflow.
- Our customized Huffman coding serves for HPC scenarios.
  - Snapshots show high similarity across consecutive timestamps.
  - So, we may only need a quasi-optimal tree for a large group of snapshots.
  - Hence, under some circumstances, tree building can be hidden.

BackUp (State-of-the-Art SIMD)

As we can see, SZ is a $\log(n)$ (linear-time) algorithm. Due to the low computational intensity, the performance is generally bounded by memory bandwidth. Our focus is to develop GPU version of SZ.

		RTX 2060S	RTX 5000	Tesla V100	Tesla A100
specification	compute (FP32 TFLOPS)	7.18	11.15	14.13	19.49
	#multiprocessor (SM)	34	48	80	108
	memory bandwidth (GB/s)	448	448	897	1555
dual-quant	absolute perf. (GB/s)	47.4 (100.0%)	63.0 (132.9%)	252.1 (531.9%)	?
	normalized (#SM)	1.39 (100.0%)	1.31 (94.2%)	3.15 (226.6%)	?
	normalized (#SM+mem.bw)	3.11 (100.0%)	2.93 (94.2%)	3.51 (112.9%)	?

Case Study of Compression Quality: Statistical Information

FIELD	SZ-1.4	cuSZ	FIELD	SZ-1.4	cuSZ
CLOUDf48	84.99	**94.18**	QSNOWf48	84.31	**93.36**
CLOUDf48.log10	84.51	87.17	QSNOWf48.log10	84.87	84.93
Pf48	84.79	84.79	QVAPORf48	84.79	84.80
PRECIPf48	85.35	92.86	TCf48	84.79	84.79
PRECIPf48.log10	84.82	84.77	Uf48	84.79	84.79
QCLOUDf48	85.03	98.91	Vf48	84.79	84.79
QCLOUDf48.log10	85.22	95.21	Wf48	84.79	84.79
QGRAUPf48	88.21	97.02	baryon_density	89.71	**98.25**
QGRAUPf48.log10	84.90	84.82	dark_matter_density	86.57	87.77
QICEf48	84.61	95.51	temperature	84.77	84.77
QICEf48.log10	85.56	85.77	velocity_x	84.77	84.77
QRAINf48	85.36	97.37	velocity_y	84.77	84.77
QRAINf48.log10	84.93	84.56	velocity_z	84.77	84.77
Hurricane avg.	85.01	86.96	Nyx avg.	85.58	85.98

Table 6: Comparison of PSNR between cuSZ and SZ-1.4 on Hurricane (FIRST 20) and Nyx (LAST 6) under valrel = 10^{-4}.

CLOUDf48

min	1%	25%	50%	75%	99%	max	range
0.00e+0	0.00e+0	0.00e+0	0.00e+0	0.00e+0	2.53e-4	2.05e-3	2.05e-3

88.5% data no greater than min+0.1*eb*

89.2% data no greater than min+*eb*

QSNOWf48

min	1%	25%	50%	75%	99%	max	range
0.00e+0	0.00e+0	1.11e-10	1.96e-9	6.34e-9	6.01e-5	8.56e-4	8.56e-4

80.9% data no greater than min+0.1*eb*

88.9% data no greater than min+*eb*

baryon density

min	1%	25%	50%	75%	99%	max	range
5.80e-2	1.37e-1	3.22e-1	5.06e-1	8.75e-1	7.42e+0	1.16e+5	1.16e+5

84.4% data no greater than min+0.1*eb*

99.5% data no greater than min+*eb*

Table 7: Statistical information (percentile) of example fields having high PSNR under valrel = 10^{-4}. The range of *eb* or even $\frac{1}{10}$*eb* at 0 or min value cover a majority of data in the fields.