SC23
Denver, CO | i am hpc.

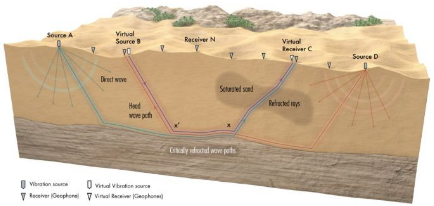
cuSZp: An Ultra-fast GPU Error-bounded Lossy Compressor with Optimized End-to-End Performance

Yafan Huang, Sheng Di, Xiaodong Yu, Guanpeng Li, Franck Cappello

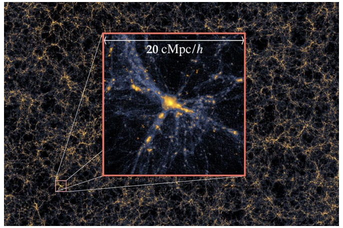
The University of Iowa
Argonne National Laboratory
Stevens Institute of Technology

Lossy Compression in HPC

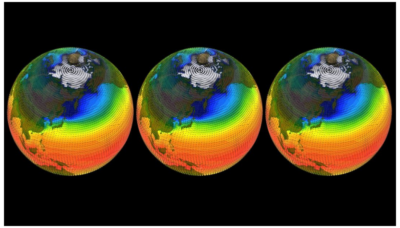
- **Lossy compression** can reduce data size drastically.
  - ***Higher compression ratio*** than lossless compression.
  - Introduced errors are controllable – ***error-bounded lossy compression***.
- **Error-bounded lossy compression** is used by various domains in HPC.



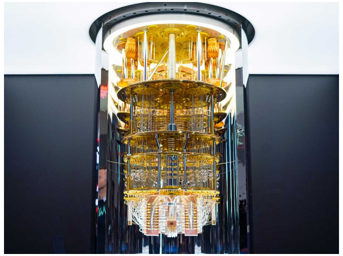
Seismic Imaging[1] *(e.g. RTM)*



Cosmology Simulation[2] *(e.g. HACC, NYX)*



Climate Simulation[3] *(e.g. CESM-ATM)*

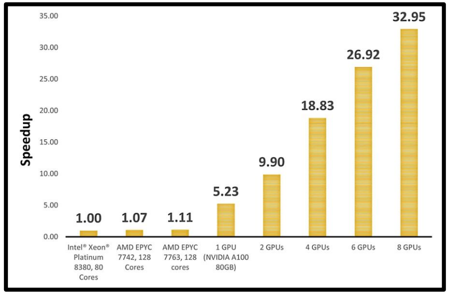


Quantum Circuit Simulation[4] *(e.g. Grover)*

1. **[Nature'2021]** Near-surface real-time seismic imaging using parsimonious interferometry
2. **[News@CMU'2021]** Machine Learning Accelerates Cosmological Simulations
3. **[TechReview@MIT'2018]** What the hell is a climate model—and why does it matter?
4. **[IEEESpectrum'2020]** IBM's concept of quantum volume tries to measure quantum computing progress in ways beyond counting qubits

GPU Lossy Compression

- GPU is critical to accelerate many **large-scale applications**.
  - CPU vs GPU performance testing in ***CFD Simulation***[1].
- Performance bottleneck: **massive data** generated.
  - ***Large memory footprint*** in GPU.
  - ***Data movement overhead*** between GPU-GPU/GPU-CPU.
- **GPU lossy compression** can address this bottleneck with fast speed.
  - Encoding phase in SZx[2]: ***~10 GB/s*** in ***CPU*** (in OpenMP), ***~200 GB/s*** in ***GPU***.



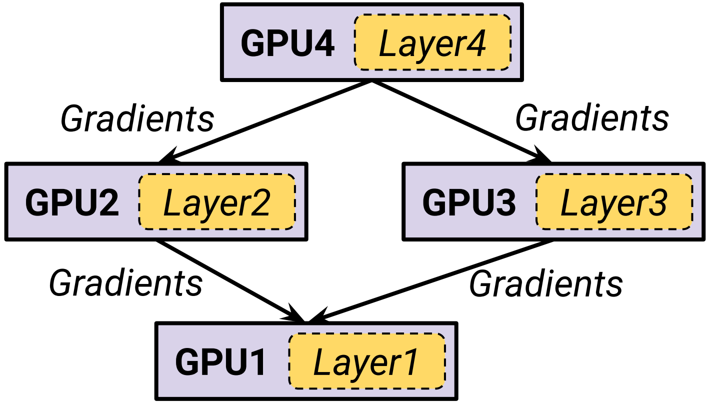
CPU vs GPU Performance in Aerodynamics Benmark[1]

A demand for using GPU Lossy Compression to accelerate large-scale GPU applications.

1. https://www.ansys.com/blog/unleashing-the-power-of-multiple-gpus-for-cfd-simulations
2. [**HPDC'2022**] Ultrafast Error-Bounded Lossy Compression for Scientific Datasets

Rethinking Use Cases for GPU Lossy Compression

- **Distributed training** for an ML model, a simple **model parallelism** case.

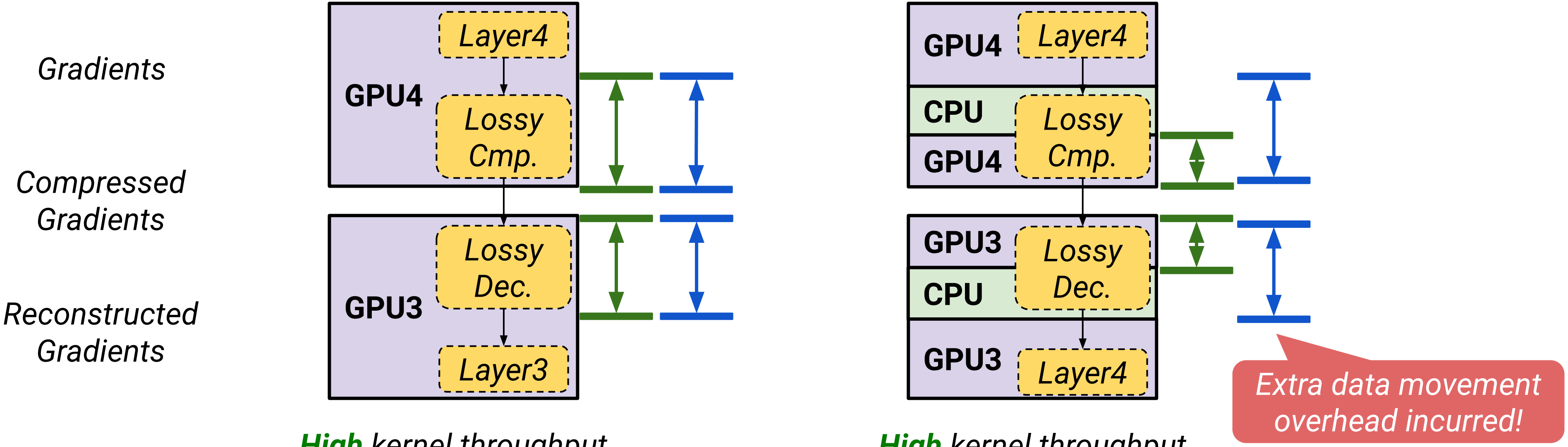


Distributed ML Model Training (One Layer for Each GPU)

- **Compressing gradients** to reduce data movement overhead across devices.

Rethinking Use Cases for GPU Lossy Compression

- How to evaluate: **End-to-end throughput** or **kernel throughput**?
  - ***Kernel***: compression-related functions that execute on GPU.
  - ***End-to-end***: input – arrays on GPU, output – arrays still on GPU.



High *kernel throughput*
High *end-to-end throughput*

High *kernel throughput*
Low *end-to-end throughput*

Limitations for Existing GPU Lossy Compressor

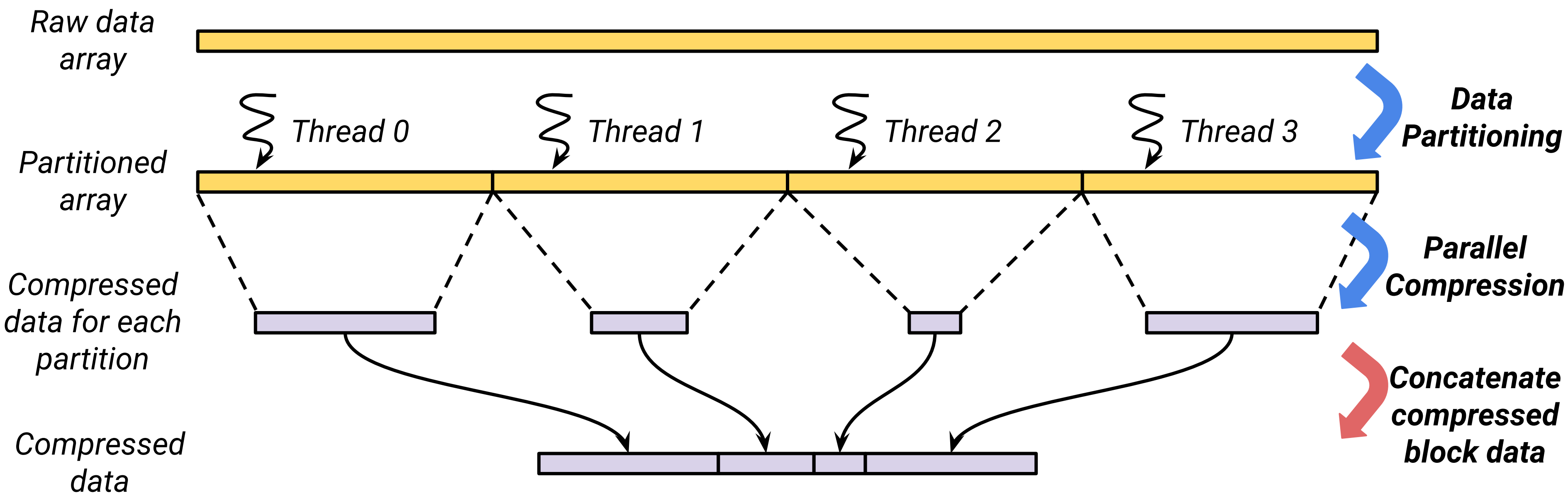
- **CPU-GPU hybrid designs** (e.g. cuSZ, cuSZx):
  - ***Pros***: Good for offline compression, good visualization quality.
  - ***Cons***: Bad for inline compression, low end-to-end throughput.
- **No error control supports** (e.g. cuZFP):
  - ***Pros***: Single kernel, high end-to-end throughput.
  - ***Cons***: No error-control supported, not enough for post-hoc analysis.

Promising solution:

Pure GPU Design	*Single GPU Kernel*	*Error Control*
High E2E/Kernel Throughput	*High CMP Ratio*	*High Data Quality*

Challenges for Designing Such a Compressor

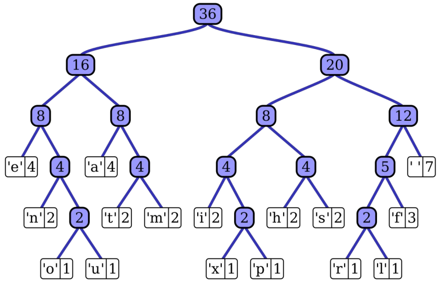
- Challenge 1: kernel fusion with **linear recurrences**.



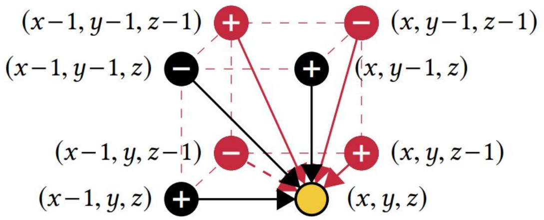
Linear recurrence exists! Index for each compressed block is based on its previous one.

Challenges for Designing Such a Compressor

- Challenge 2: balancing **high throughput** and **high cmp. ratio & data quality**.
  - Fancier algorithms, more computation, lower throughput.
  - Kernel fusion for linear recurrences even complicates the compression algorithm.



Building A Huffman Tree[1]



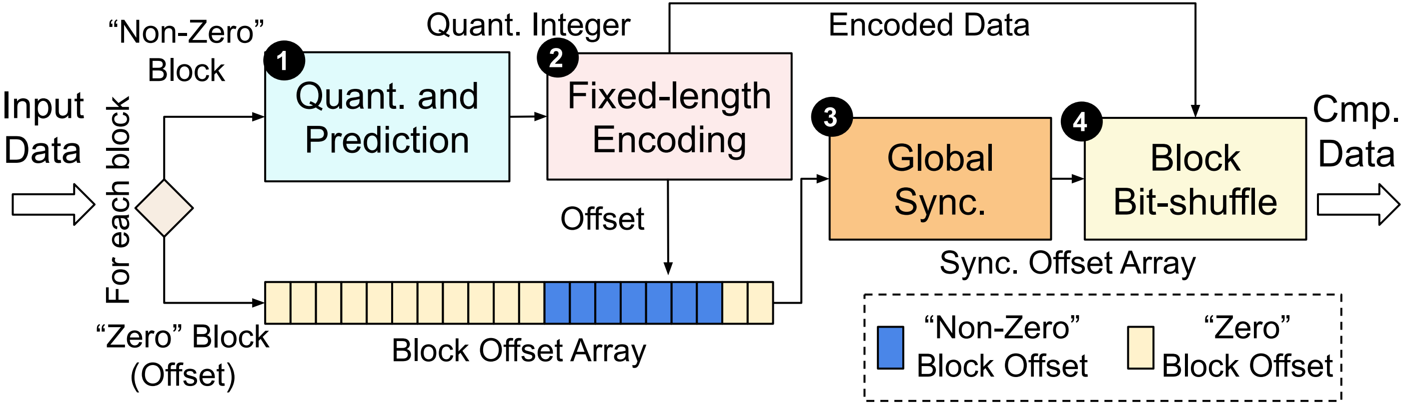
3D Lorenzo Prediction[2]

It’s hard to both have your cake and eat it.

1. https://en.wikipedia.org/wiki/Huffman_coding
2. **[PPoPP’2020]** waveSZ: a hardware-algorithm co-design of efficient lossy compression for scientific data

Our Solution: cuSZp

- **cuSZp** is an error-bounded GPU lossy compressor with **single kernel function**.



High-level Overview of cuSZp Compression Kernel

High Throughput

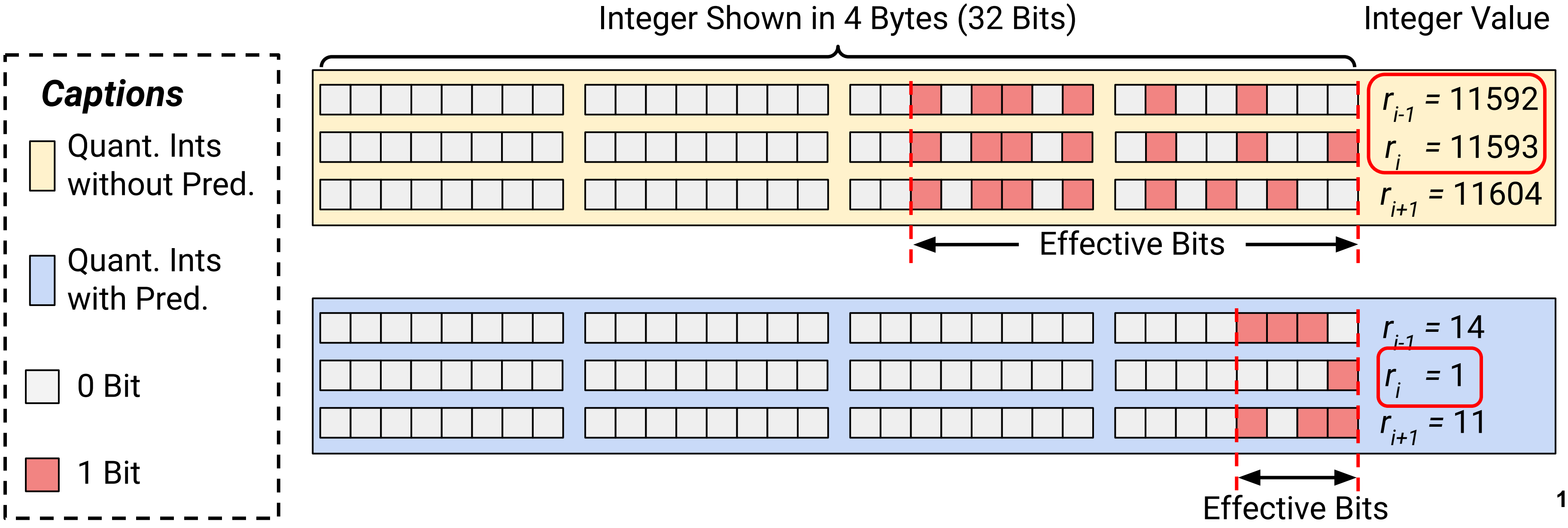
High Compression Ratio

Error Control Supported

cuSZp: Quantization and Prediction

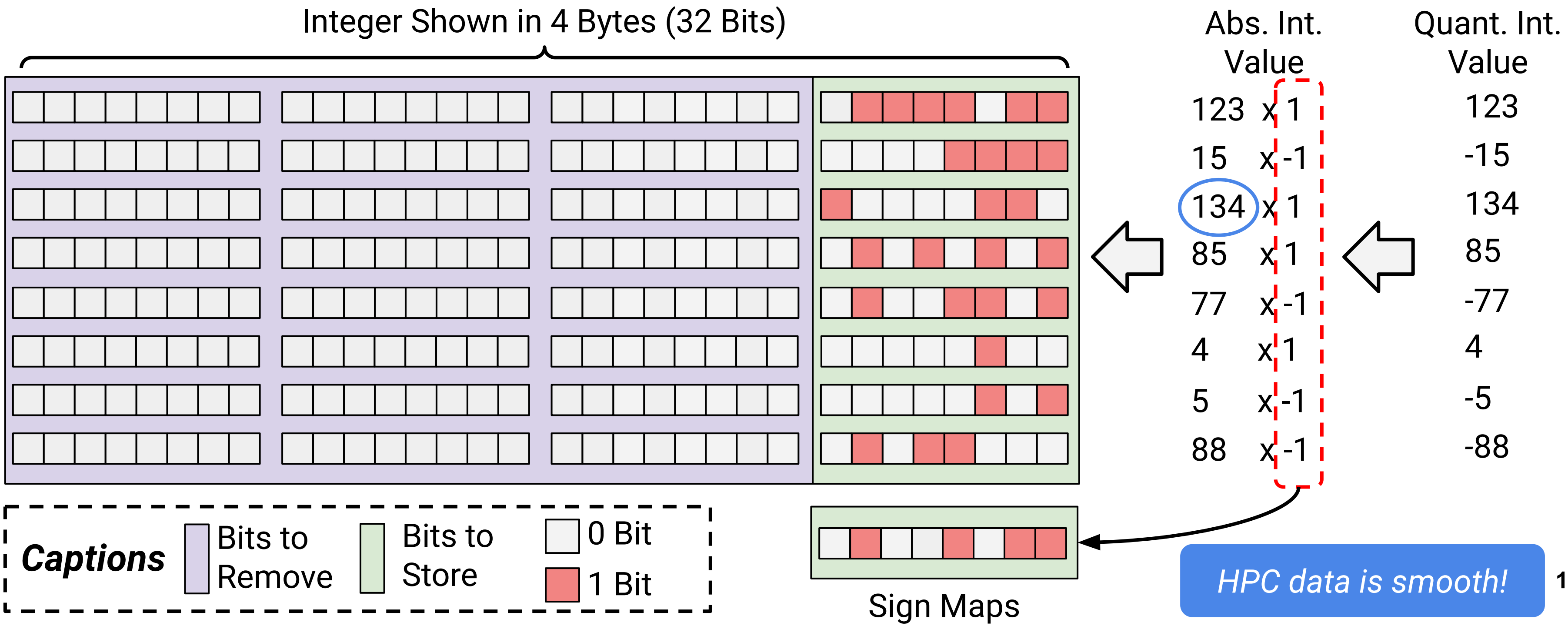
The only "Lossy" step in cuSZp.

- **Quantization** converts a floating point number into an integer.
  - Example: *eb = 0.01, fp data = 17.335 -> quantization integer = 867.*
- **Prediction** removes redundant bit patterns.



cuSZp: Fixed-length Encoding

- **Fixed-length encoding** preserves a fixed amount of bit for each integer.
  - The fixed amount is determined by ***the greatest value*** for each data block.



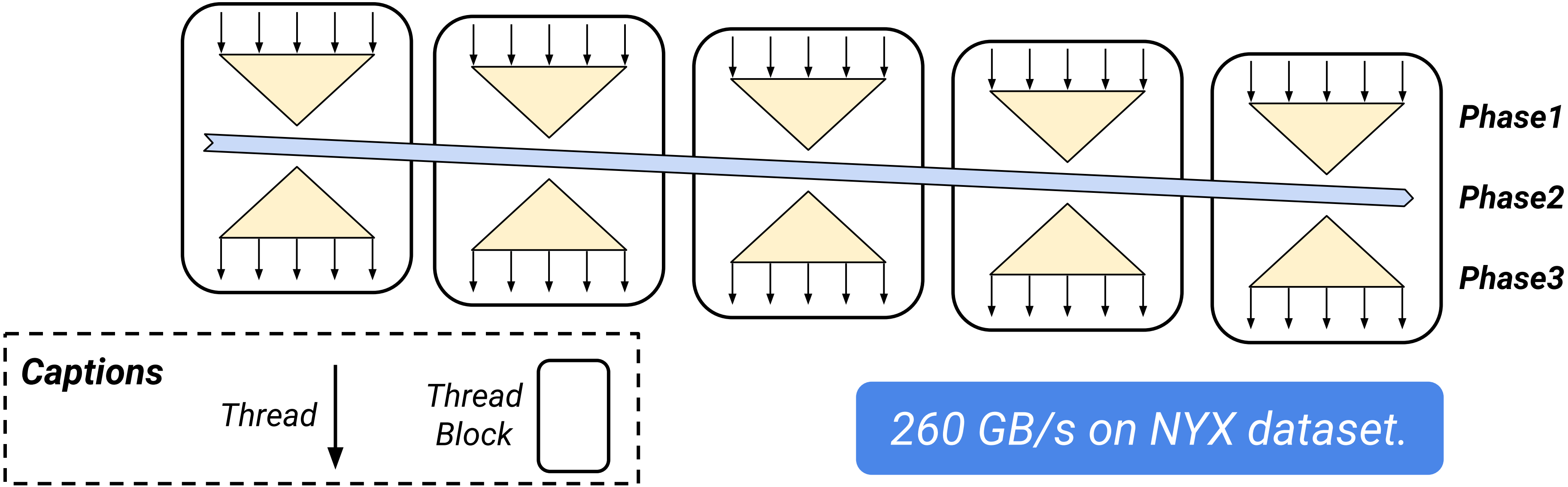
cuSZp: Global Synchronization

- **Global Synchronization** generates index for each compressed block.
  - Phase 1: prefix-sum inside each threadblock.
  - Phase 2: inter-threadblock synchronization via ***single pass chained-scan.***
  - Phase 3: restore global prefix-sum inside each threadblock.

Thread Blk.

Global

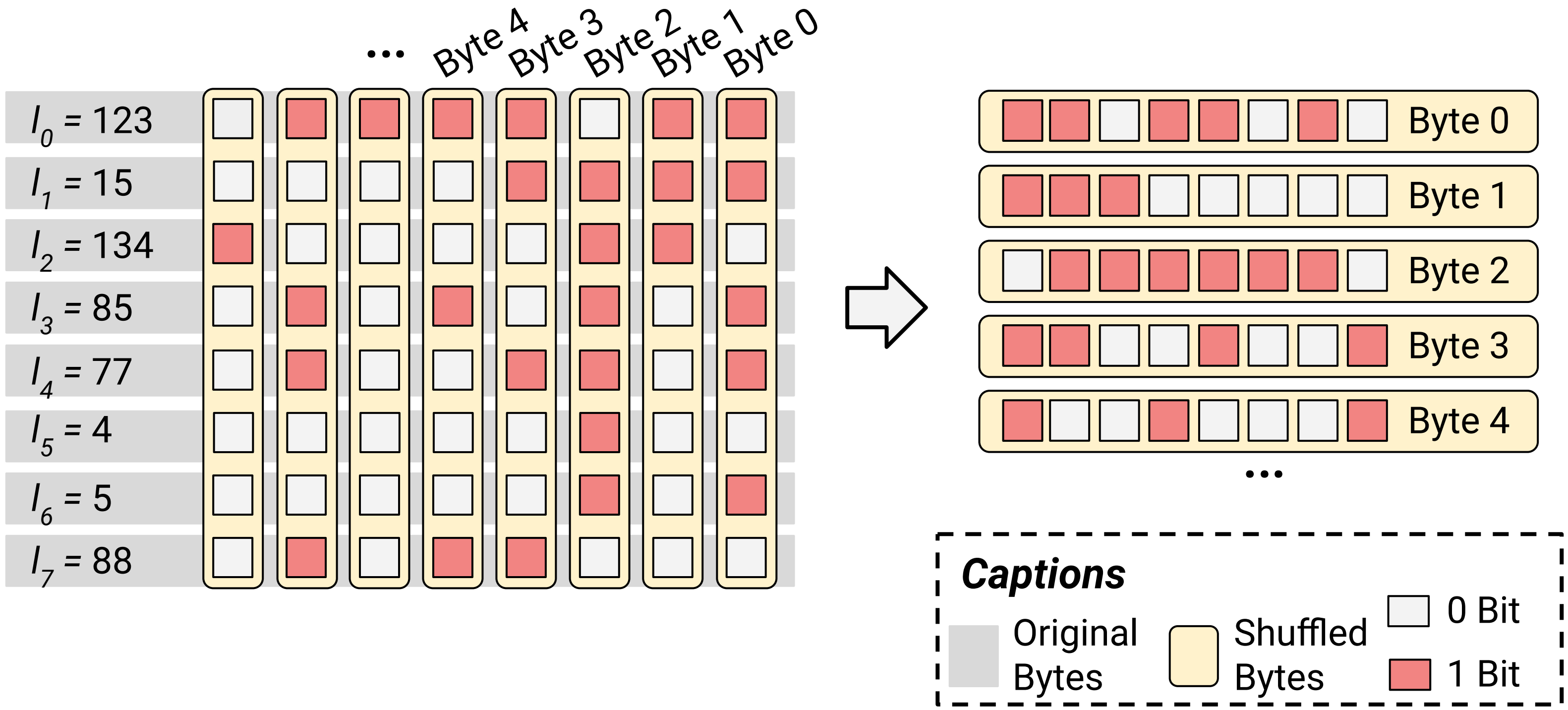
Thread Blk.



260 GB/s on NYX dataset.

cuSZp: Block Bit-shuffle

- **Block bit-shuffle** does not modify compressed data – it rearranges data to make this process suitable for accessing global memory in a parallel manner.



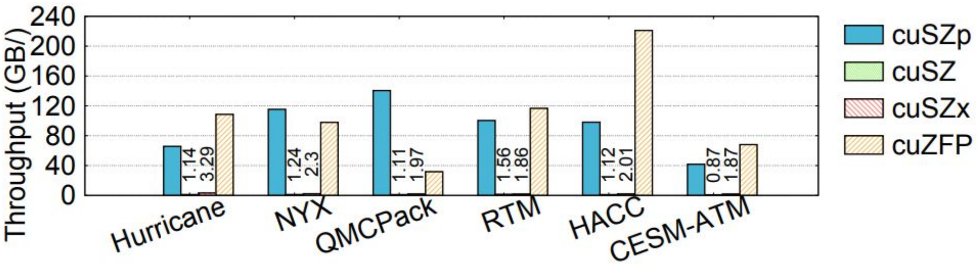
Evaluation: Settings

- Argonne NATIONAL LABORATORY **Swing Cluster**
  - NVIDIA A100 GPU (40GB)
  - AMD EPYC 7742 CPUs
  - 1 TB DDR4 Memory
- **Baseline Compressor**
  - *cuSZ*[1]
  - *cuSZx*[2]
  - *cuZFP*[3]
- **Evaluation Metrics**
  - Throughput (GB/s)
  - Compression ratio
  - Reconstructed data quality
- **HPC Datasets**
  - *Hurricane*: weather simulation
  - *NYX*: cosmology simulation
  - *QMCPack*: quantum computing
  - *RTM*: seismic imaging
  - *HACC*: cosmic simulation
  - *CESM-ATM*: climate simulation

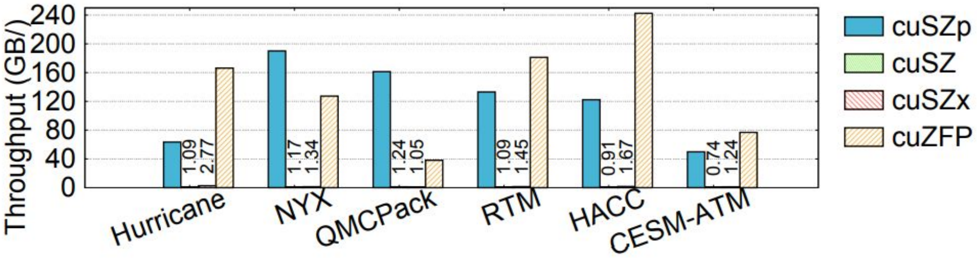
1. [**PACT'2020**] cuSZ: An Efficient GPU-Based Error-Bounded Lossy Compression Framework for Scientific Data
2. [**HPDC'2022**] Ultrafast Error-bounded Lossy Compression for Scientific Datasets
3. [**TVCG'2014**] Fixed-Rate Compressed Floating-Point Arrays

Evaluation: End-to-End Throughput

- **End-to-End Throughput:** input array on GPU, output array on GPU.



(a) End-to-end compression throughput.

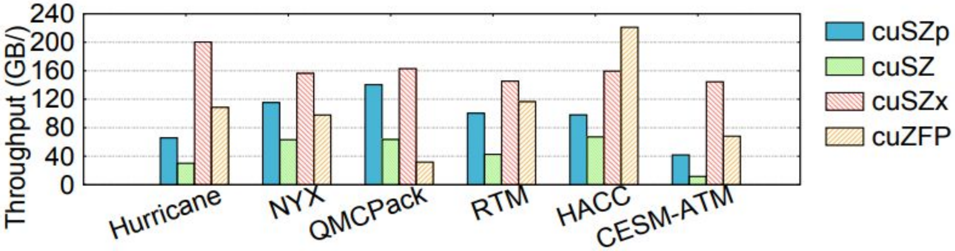


(b) End-to-end decompression throughput.

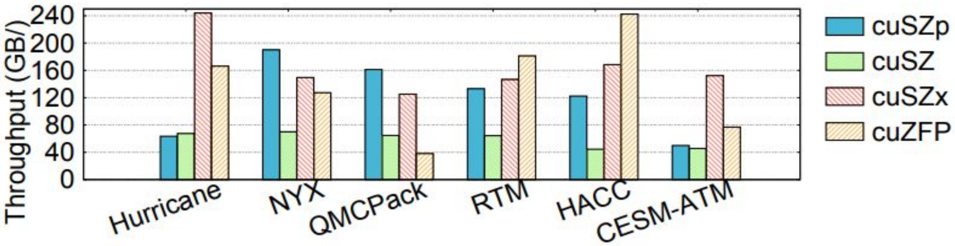
- **95.53x** with cuSZ and **55.18x** with cuSZx, due to their CPU-GPU hybrid design.

Evaluation: Kernel Throughput

- **Kernel Throughput:** GPU computation kernel throughput.



(a) Kernel compression throughput.



(b) Kernel decompression throughput.

- **93.63 GB/s** and **120.04 GB/s** for compression/decompression throughput.

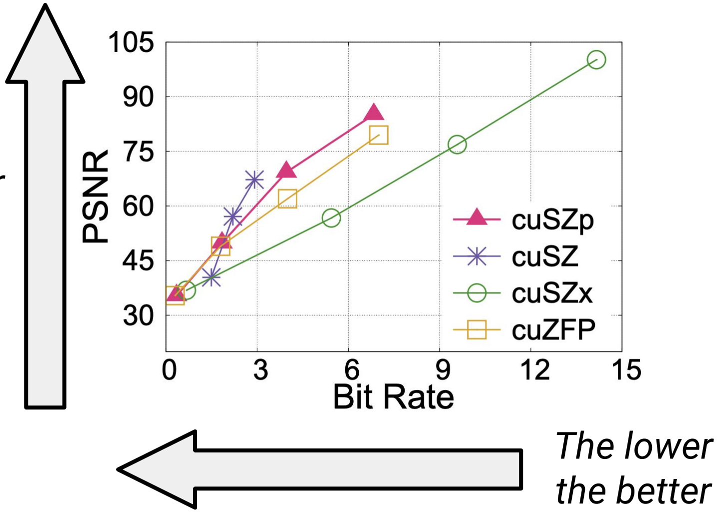
Evaluation: Compression Ratio

		Hurricane			NYX			QMCPack			RTM			HACC			CESM-ATM		
	REL	min	max	avg	min	max	avg	min	max	avg	min	max	avg	min	max	avg	min	max	avg
cuSZp	1E-1	13.56	124.32	75.45	43.48	127.99	99.11	85.20	98.25	91.73	72.76	127.99	108.48	10.35	59.82	34.30	3.99	101.27	27.40
	1E-2	5.96	88.88	38.71	9.62	127.80	66.74	12.46	22.23	17.35	13.89	127.96	67.06	5.24	10.09	7.63	2.93	43.75	14.21
	1E-3	3.72	56.88	22.32	5.10	125.55	38.46	6.08	10.08	8.08	6.88	127.83	42.40	3.43	5.20	4.31	2.31	33.81	9.82
	1E-4	2.71	36.66	14.36	3.36	98.25	22.15	3.79	5.57	4.68	4.16	127.59	27.56	2.53	3.39	2.96	1.81	26.11	7.35
cuSZ	1E-1	26.42	29.98	28.73	31.24	31.58	31.47	19.41	23.41	21.41	29.47	30.87	30.45	30.31	31.30	30.81	23.31	25.43	24.63
	1E-2	15.35	28.62	22.53	28.71	31.57	30.22	7.50	21.55	14.53	n/a	n/a	n/a	n/a	n/a	n/a	19.18	25.33	22.89
	1E-3	8.91	23.61	15.97	n/a	n/a	n/a	4.26	17.70	10.98	n/a	n/a	n/a	n/a	n/a	n/a	11.34	25.16	18.48
	1E-4	3.37	17.25	8.36	10.75	31.28	16.22	n/a	n/a	n/a	3.67	30.84	11.63	n/a	n/a	n/a	5.38	24.43	12.47
cuSZx	1E-1	28.68	118.27	74.19	77.09	124.10	110.74	25.59	69.21	47.40	23.36	124.06	76.69	28.81	124.08	70.41	16.05	124.11	74.30
	1E-2	3.91	53.92	21.67	4.68	123.72	61.43	2.74	9.01	5.88	3.94	123.92	37.51	3.05	117.57	44.37	3.93	124.11	31.85
	1E-3	2.86	32.03	13.47	3.11	118.93	30.37	2.36	4.31	3.34	2.83	123.55	23.74	2.18	4.31	3.00	2.77	123.96	24.24
	1E-4	2.03	23.64	10.29	2.38	74.36	15.12	1.68	2.84	2.26	2.17	123.00	18.46	1.70	2.68	2.13	2.11	123.77	22.57

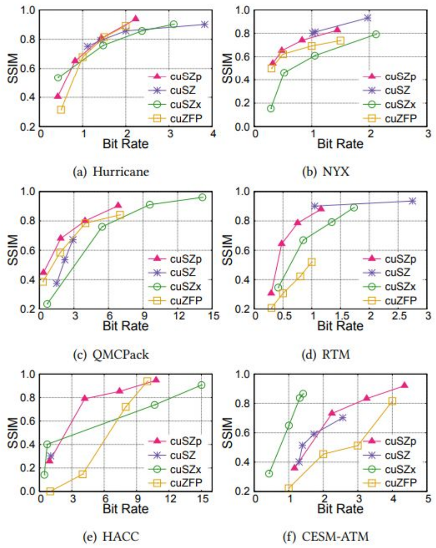
- cuSZp achieves **higher compression ratio on 16/24 cases** than cuSZ and cuSZx.
- cuZFP is not evaluated here due to different designs.

Evaluation: Data Quality - Rate Distortion

- **Rate Distortion:** PSNR/SSIM unders the same compression ratio (bit rate).
  - ***Bit rate***: how many bits are used to store one floating point data.
  - ***PSNR & SSIM***: quantitative metrics for evaluating reconstructed data quality.



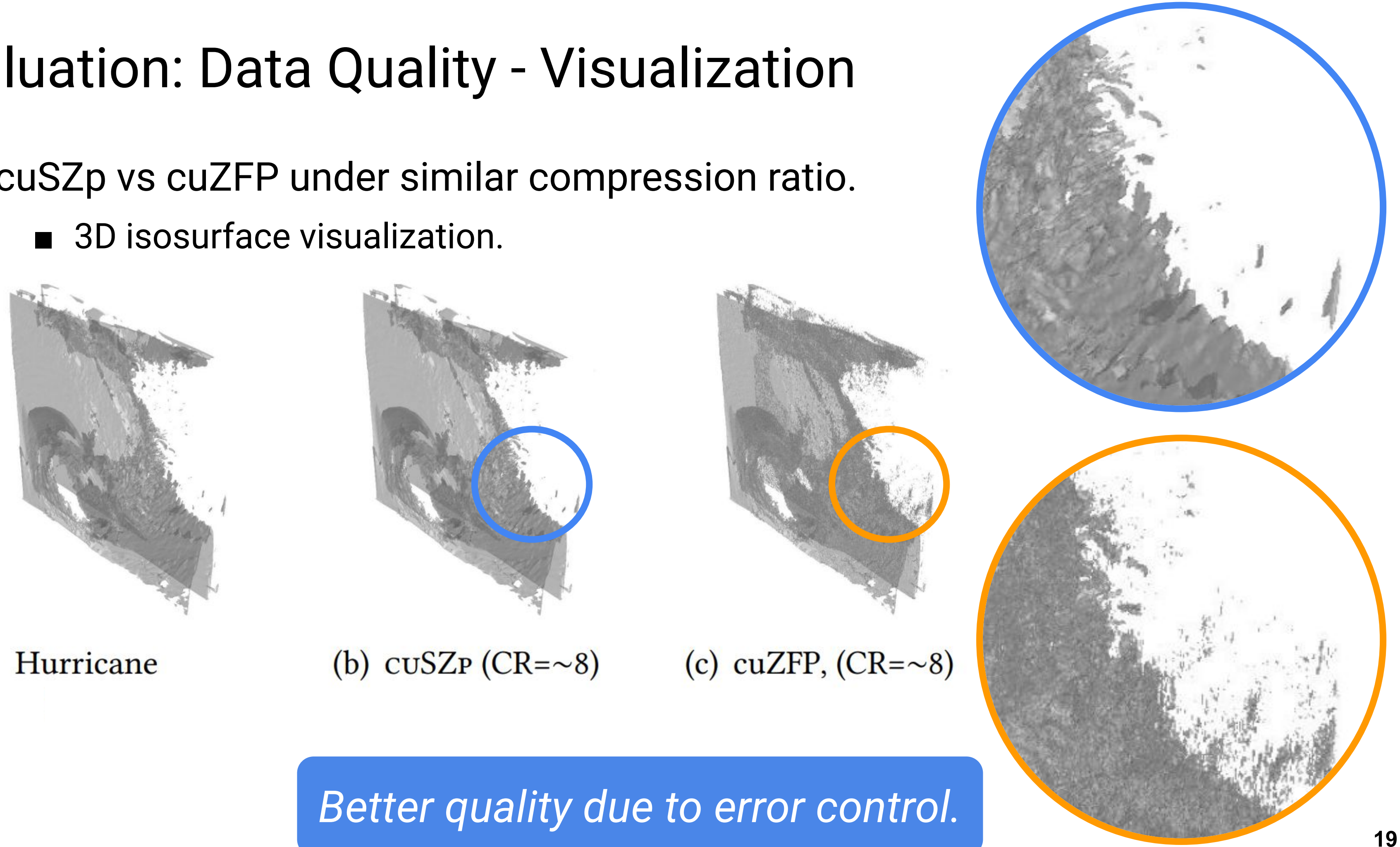
Rate Distortion (PSNR)



Rate Distortion (SSIM)

Evaluation: Data Quality - Visualization

- cuSZp vs cuZFP under similar compression ratio.
  - 3D isosurface visualization.



(a) Hurricane (b) cuSZp (CR=~8) (c) cuZFP, (CR=~8)

Better quality due to error control.

Code Example: How to Use cuSZp

```
#include <cuSZp_utility.h>
#include <cuSZp_entry_f32.h>

// For measuring the end-to-end throughput.
TimingGPU timer_GPU;

// cuSZp compression.
timer_GPU.StartCounter(); // set timer
SZp_compress_deviceptr_f32(d_oriData, d_cmpBytes,
                           nbEle, &cmpSize, errorBound, stream);
float cmpTime = timer_GPU.GetCounter();

// cuSZp decompression.
timer_GPU.StartCounter(); // set timer
SZp_decompress_deviceptr_f32(d_decData, d_cmpBytes,
                             nbEle, cmpSize, errorBound, stream);
float decTime = timer_GPU.GetCounter();
```

Single-precision cuSZp compression/decompression (device pointers)

Summary

- cuSZp is a **single kernel** error-bounded lossy compressor on GPU.
- **93.63 GB/s** and **120.04 GB/s** for compression and decompression.
- cuSZp also achieves **high compression ratio** and **high data quality**.
- cuSZp now supports both **single- & double-precision floating point data**.
- cuSZp also performs well on **lower-end GPUs** (e.g. RTX 3080).
- Open source: https://github.com/szcompressor/cuSZp/

Yafan Huang
University of Iowa
yafan-huang@uiowa.edu
https://hyfshishen.github.io

THE UNIVERSITY OF IOWA ORGANIZED FEB. 25TH 1847.

Argonne NATIONAL LABORATORY