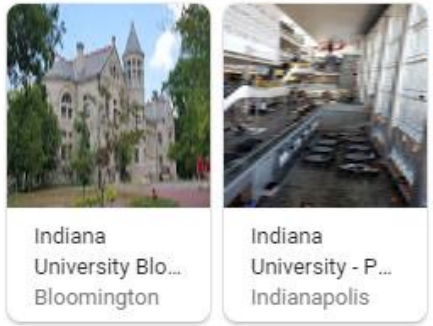
INDIANA UNIVERSITY
BLOOMINGTON

Advancing HPC I/O and Storage via Efficient Data Compression

Dingwen Tao
Indiana University
July 4, 2022

HiPDAC

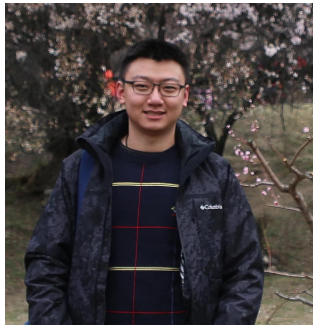

Indiana University Blo...
Bloomington

Indiana University - P...
Indianapolis


Purdue University
West Lafayette


University of Notre Dame
Notre Dame

Graduate Students



Undergraduate Students

Thank You!

Research topics (not limited to):

- Big data management, analytics, visualization
- Large-scale machine/deep learning
- Heterogeneous computing (GPU/FPGA)
- Fault tolerance and resilience at extreme scale
- Energy-efficient computing
- Numerical algorithms, simulation & software

Storage and I/O Issues in HPC Systems

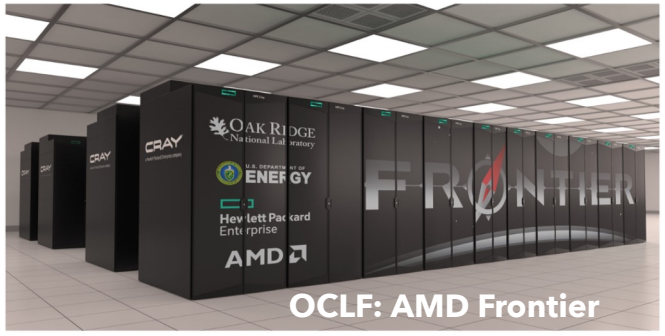
SUPERCOMPUTER SYSTEM	YEAR	CLASS	PEAK FLOPS (PF)	MEMORY SIZE (MS)		STORAGE BAND -WIDTH (SB)		MS/SB	PF/SB
Cray Jaguar	2008	1 PFLOPS	1.75 PFLOPS	360 TB		240 GB/s		1.5k	7.3k
Cray Blue Waters	2012	10 PFLOPS	13.3 PFLOPS	1.5 PB		1.1 TB/s		1.3k	13.3k
Cray CORI	2017	10 PFLOPS	30 PFLOPS	1.4 PB		1.7 TB/s	(**)	0.8k	17k
IBM Summit	2018	100 PFLOPS	200 PFLOPS	> 10 PB	(*)	2.5 TB/s		> 4k	80k

(*) when using burst buffer (**) counting only DDR4 source: F. Cappello (ANL)

The compute capability is ever-growing, but storage capacity and bandwidth are developing much **more slowly**

SUPERCOMPUTER SYSTEM	YEAR	CLASS	PEAK FLOPS (PF)	MEMORY SIZE (MS)		STORAGE BAND -WIDTH (SB)		MS/SB	PF/SB
Fujitsu Fugaku	2020	"ExaScale"	537 PFLOPS	(*) 4.85 PB		> 1.5 TB/s	(**)	> 3.23k	358k
AMD Frontier	2021	ExaScale	1.6 EFLOPS	9.2 PB	(a)	10 TB/s		> 0.92k	160k
Intel Aurora (#)	future	ExaScale	> 2 EFLOPS	> 10 PB	(a)	>= 25 TB/s		> 0.40k	80k

(*) Rpeak, Top-500 as of November 2020 (**) DDN Newsroom
(a) aggregated memory (CPU DDR + GPU HBM)



Trend of HPC Systems: Heterogeneity

LUDDY
SCHOOL OF INFORMATICS, COMPUTING, AND ENGINEERING

Rank (Prev.)		Name	Year	CPU Cores	Accelera-tor Cores	Rmax [PFlop/s]	Rpeak [PFlop/s]	Interconnect	Manufac-turer	Country & Site
1		Frontier	2021	8,730,112	8,138,240	1,102.0	1,685.7	Slingshot-11	HPE	United States; DOE/SC/Oak Ridge National Laboratory
2	(1)	Fugaku	2020	7,630,848	0	442.0	537.2	Tofu interconnect D	Fujitsu	Japan; RIKEN Center for Computational Science
3		LUMI	2022	1,110,144	1,034,880	151.9	214.4	Slingshot-11	HPE	Finland; EuroHPC/CSC
4	(2)	Summit	2018	2,414,592	2,211,840	148.6	200.8	Infiniband EDR	IBM	United States; DOE/SC/Oak Ridge National Laboratory
5	(3)	Sierra	2018	1,572,480	1,382,400	94.6	125.7	Infiniband EDR	IBM/NVIDIA	United States; DOE/NNSA/LLNL
6	(4)	Sunway TaihuLight	2016	10,649,600	0	93.0	125.4	Sunway	NRCPC	China; National Supercomputing Center in Wuxi
7	(5)	Perlmutter	2021	761,856	663,552	70.9	93.8	Slingshot-10	HPE	United States; DOE/SC/LBNL/NERSC
8	(6)	Selene	2020	555,520	483,840	63.5	79.2	Infiniband HDR	Nvidia	United States; NVIDIA Corporation
9	(7)	Tianhe-2A	2018	4,981,760	4,554,752	61.4	100.7	TH Express-2	NUDT	China; National Super Computer Center in Guangzhou
10		Adastra	2022	319,072	297,440	46.1	61.6	Slingshot-11	HPE	France; GENCI-CINES

More and more **heterogeneous** systems

- CPU + GPU (80% in TOP 10)
- Memory/storage hierarchy
- SmartNIC (FPGA, DPU)
- ...

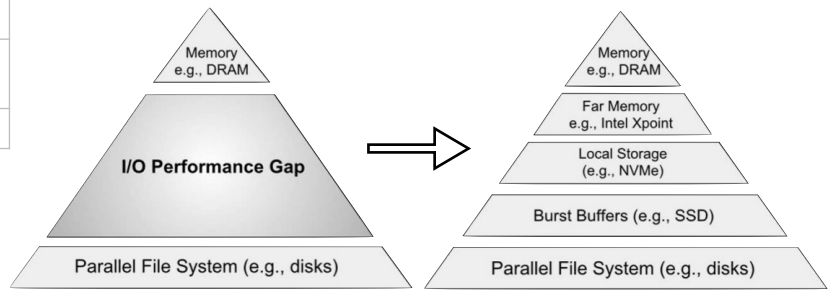
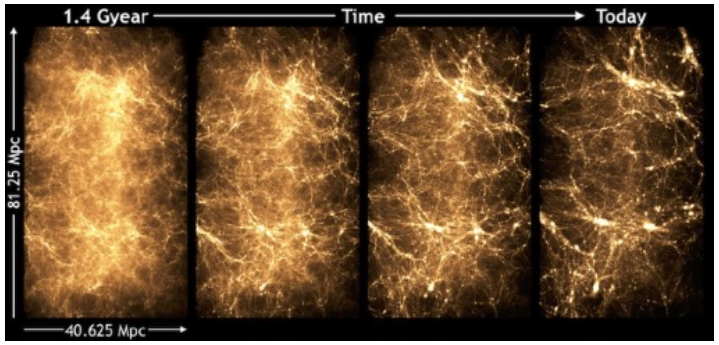


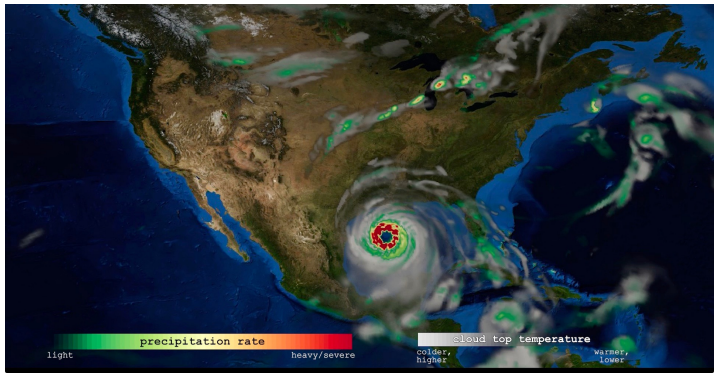
Figure from X. Sun (IIT)

Data Management Issues for Scientific Applications

LUDDY
SCHOOL OF INFORMATICS, COMPUTING, AND ENGINEERING

application	data scale	bottleneck	reduce by
HACC cosmology simulation	**20 PB** one-trillion-particle	**use up filesystem (26 PB in total)** Mira@ANL	**10×** in need
CESM climate simulation	**50%** vs 20% storage in hardware budget, **2017** vs 2013	**5h30m to store** NSF Blue Waters 1-TBps I/O	**10×** in need
APS-U High-Energy X-Ray Beams Experiments	**10^2 PB** Brain Initiatives	**saturate connection** 100 GBps bandwidth	**100×** in need





Our Solution – Error-Bounded Lossy Compression

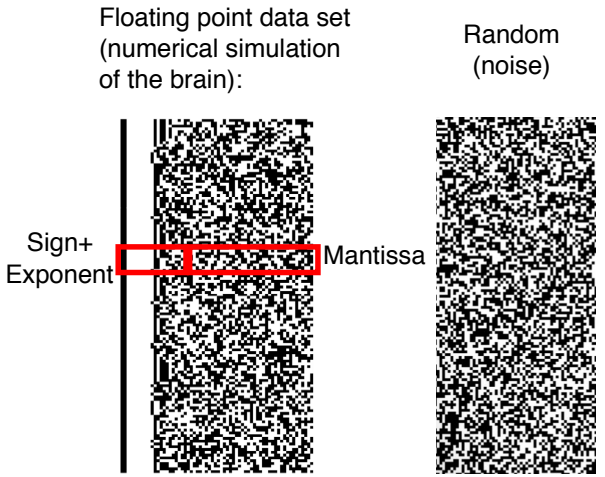
LUDDY
SCHOOL OF INFORMATICS, COMPUTING, AND ENGINEERING

2:1 (FP-type)	**10:1 or higher**
lossless on scientific datasets	reduction ratio in need
industry lossy compressor (JPEG)	high in reduction rate, but **not** suitable for **HPC**
need **diverse** compression modes	1) absolute error bound (infinity-norm) 2) pointwise relative error bound 3) RMSE error bound (2-norm) 4) fixed bitrate 5) satisfying post-analysis requirements

SZ

Di and Cappello 2016, Tao *et al.* 2017, Xin *et al.* 2018, Tian *et al.* 2020

> prediction-based lossy compressor framework for scientific data
> strictly control the global upper bound of compression error
> implemented on CPU, GPU, FPGA
> integrated in I/O libraries (HDF5, ADIOS, PnetCDF)



Source: Leonardo Bautista Gomez (BSC)

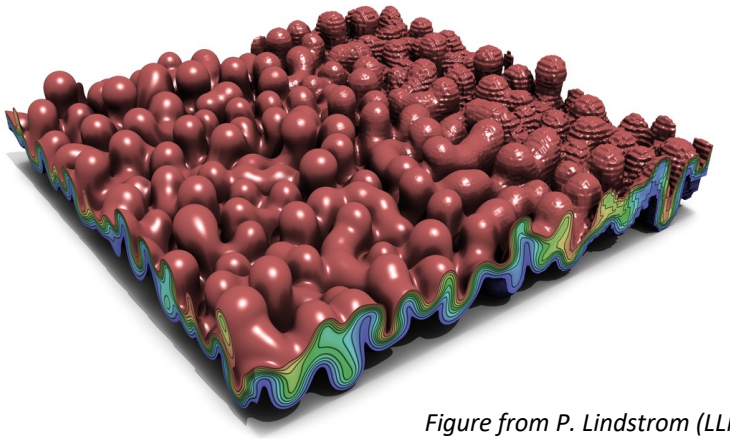


Figure from P. Lindstrom (LLNL)

Lossy compression for scientific data at varying reduction ratio (10:1 to 250:1, left to right)

Lossy Compression Does Improve Performance!

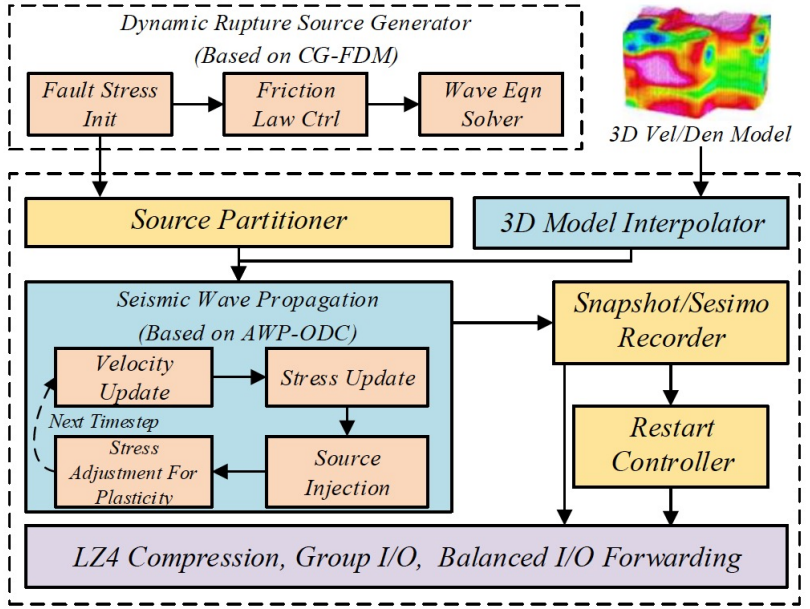
LUDDY
SCHOOL OF INFORMATICS, COMPUTING, AND ENGINEERING

2017 Gordon Bell Award: 18.9-Pflops Nonlinear Earthquake Simulation on Sunway TaihuLight: Enabling Depiction of 18-Hz and 8-Meter Scenarios

Designed a lossy compression scheme: On-The-Fly (OTF) compression

Benefit from lossy compression:

- **24%+** computational performance improved
- **2X** maximum problem size that can be solved

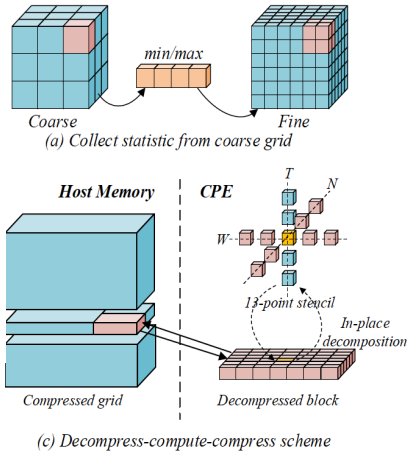


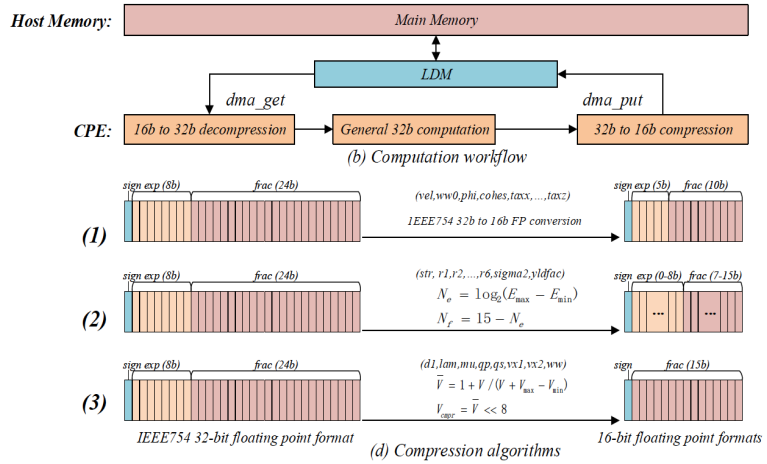
On-the-Fly compression, explored 3 methods:

M1: Directly conversion to half precision IEEE 754 standard. However, dynamic is too large for 5 bits of exponent, for some variables.

M2: Determines the required exponent bit-width according to the recorded maximum dynamic range and uses the rest bits for mantissa.

M3: Normalize all the values of the same array to the range between 1 and 2, which corresponds to an exponent value of zero.





SZ: A Lossy Compression Framework for Scientific Data

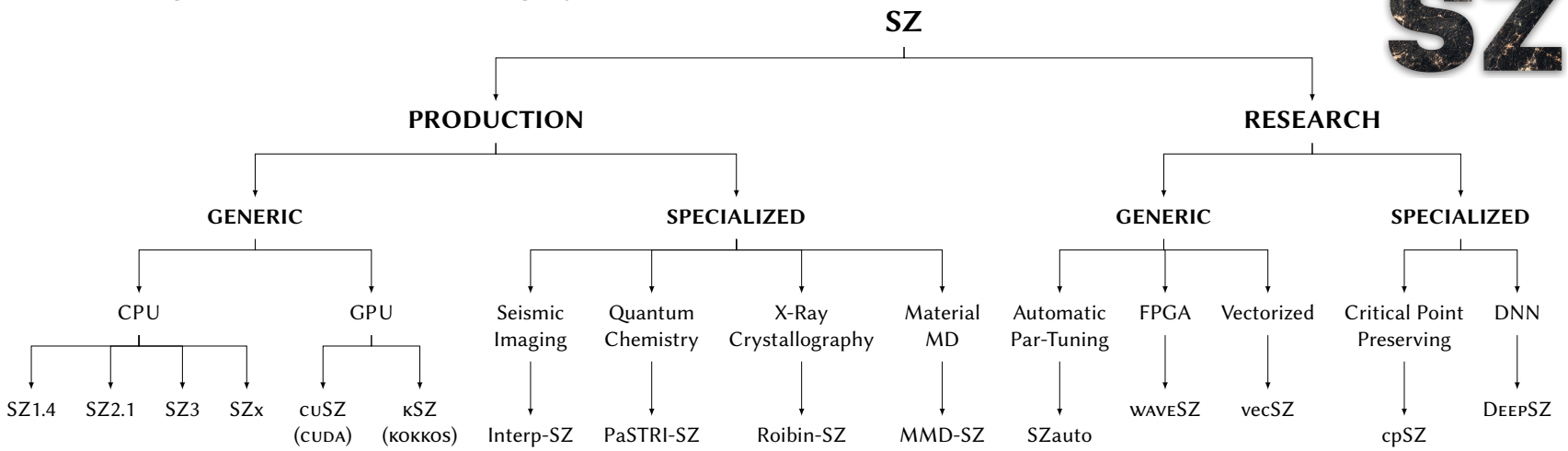
LUDDY
SCHOOL OF INFORMATICS, COMPUTING, AND ENGINEERING

$6M from DOE
$1M from NSF
$1M from Aramco

2021 R&D 100 WINNER

Established in 1963, the R&D 100 Awards is the only S&T (science and technology) awards competition that recognizes new commercial products, technologies and materials for their technological significance that are available for sale or license. The R&D 100 Awards have long been a benchmark of excellence for industry sectors as diverse as telecommunications, high-energy physics, software, manufacturing, and biotechnology. This 2021 R&D 100 winner is listed below, along with its respective category.

SZ



SZ compression framework family tree.

Core R&D Team

Argonne National Laboratory
Franck Cappello (lead), **Sheng Di** (lead)

Washington State University
Dingwen Tao (lead), Jiannan Tian, Sian Jin, Chengming Zhang, Cody Rivera

University of California, Riverside
Xin Liang (lead), Kai Zhao, Jinyang Liu

Clemson University
Jon Calhoun (lead), Robert Underwood, Griffin Dube

https://github.com/szcompressor

HPC use-cases:

- Reducing storage footprint
- Accelerating I/O & communication
- Accelerating visualization
- Reducing streaming intensity
- Running larger problems
- Checkpoint/restart

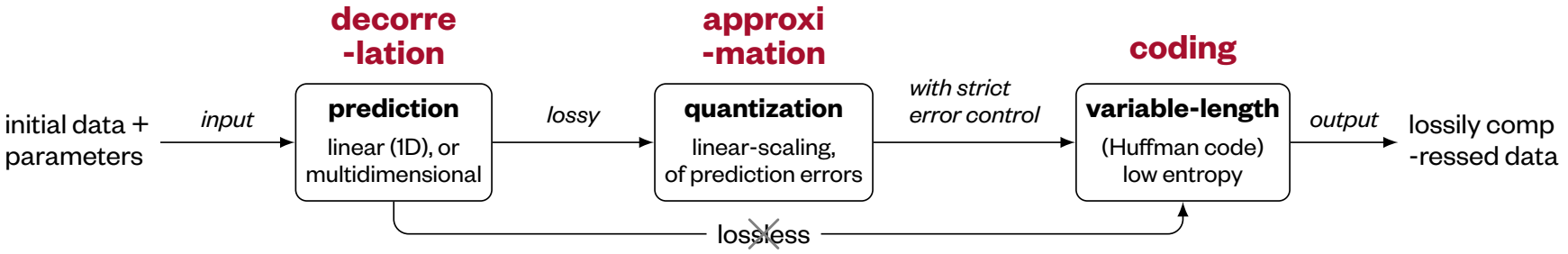
AI use-cases:

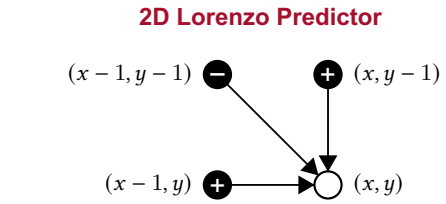
- DNN model compression
- DNN training data compression
- Reducing DNN memory consumption
- Accelerating distributed training
- …

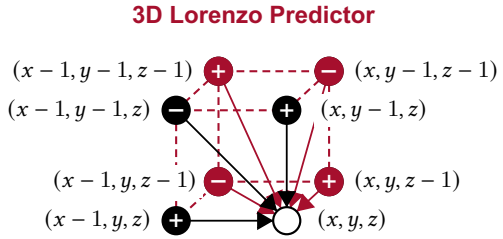
Significantly Improving Lossy Compression for Scientific Data Sets Based on Multidimensional Prediction and Error-Controlled Quantization

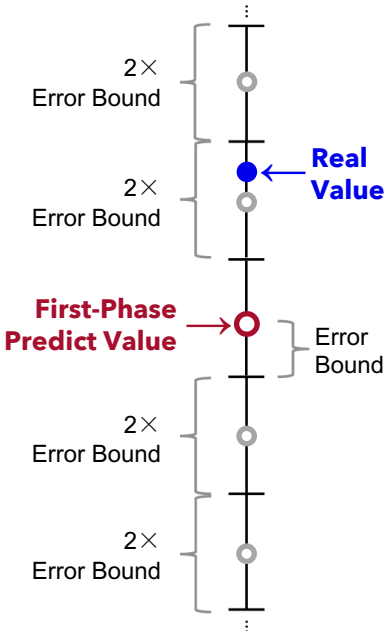
Published in *2017 IEEE International Parallel and Distributed Processing Symposium (IPDPS'17)*

SZ Compression Pipeline

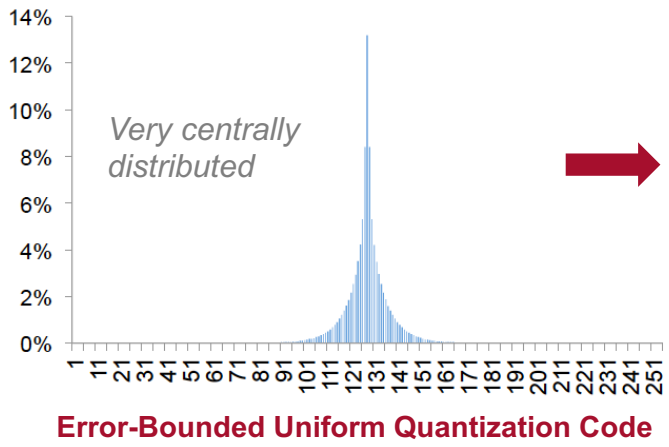


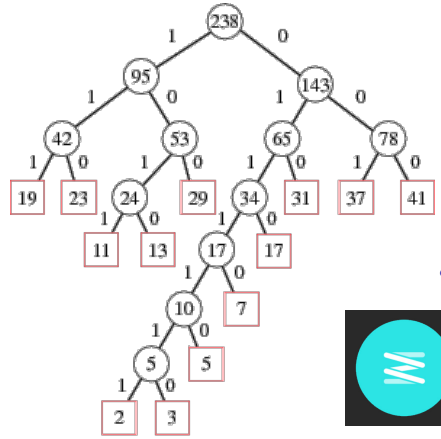






We may need much less than 256 intervals (that 8 bits can represent).





gzip

Zstandard

prediction **quantization** **coding**

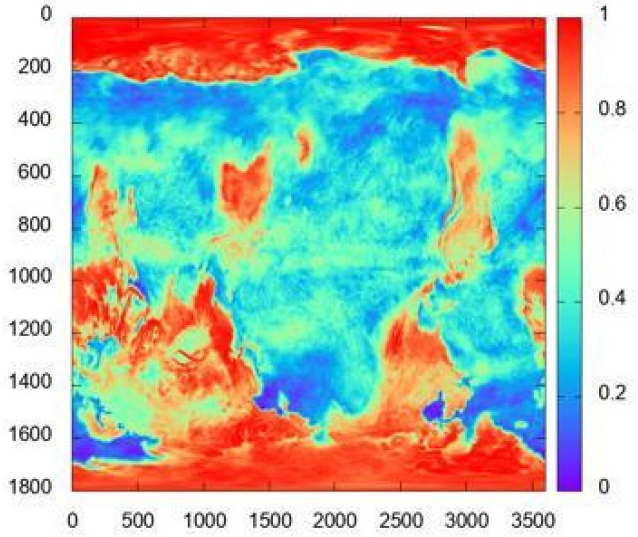
Climate and Severe Weather Datasets

LUDDY
SCHOOL OF INFORMATICS, COMPUTING, AND ENGINEERING

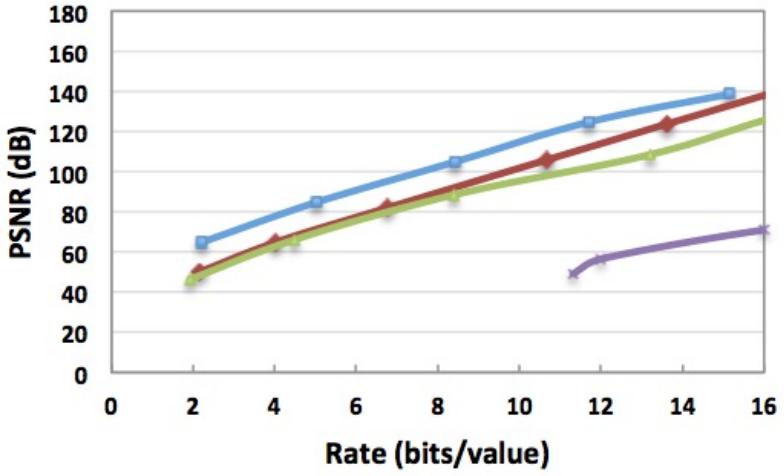
Experimental data (floating point-single precision-FP32)

- Climate: ATM (CESM): 3D dataset from climate simulation
- Weather: hurricane: 3D dataset from Hurricane Isabel simulation

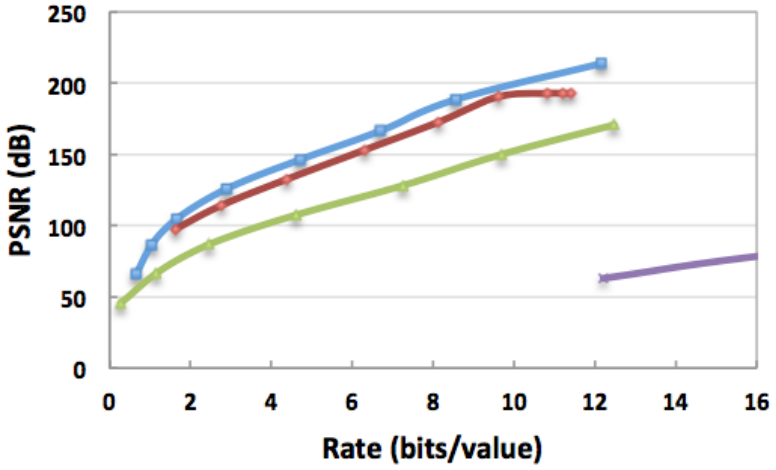
	Data Source	Dimension Size	Data Size	File Number
ATM	Climate simulation	1800 × 3600	2.6 TB	11400
APS	X-ray instrument	2560 × 2560	40 GB	1518
Hurricane	Hurricane simulation	100 × 500 × 500	1.2 GB	624

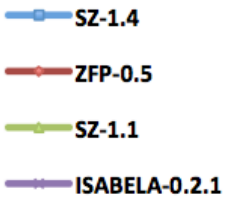


Climate



Hurricane





ZFP: Best mode "fixed-accuracy"
E.g., bit-rate = 8 bits/value (CR = 4)

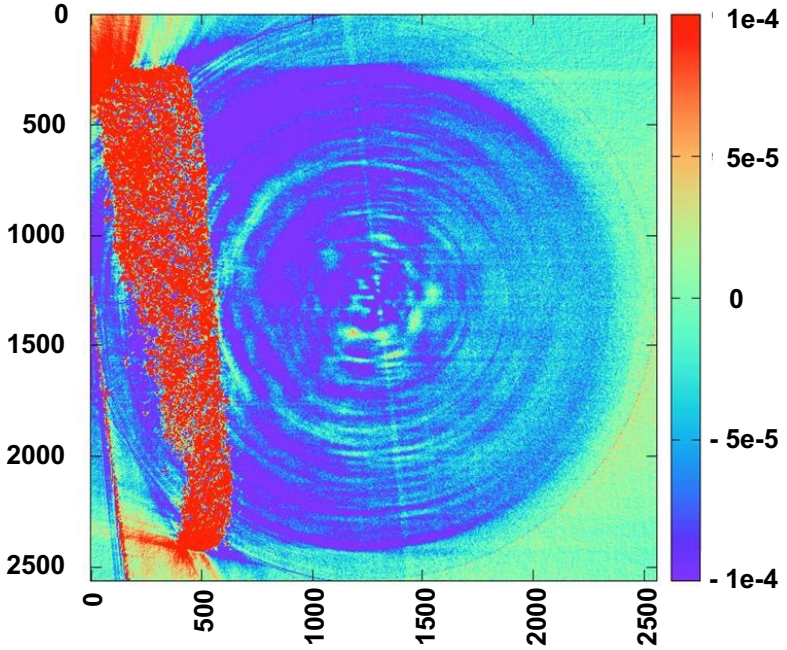
- SZ: 14dB higher than ZFP on ATM
- SZ: 11dB higher than ZFP on Hurricane

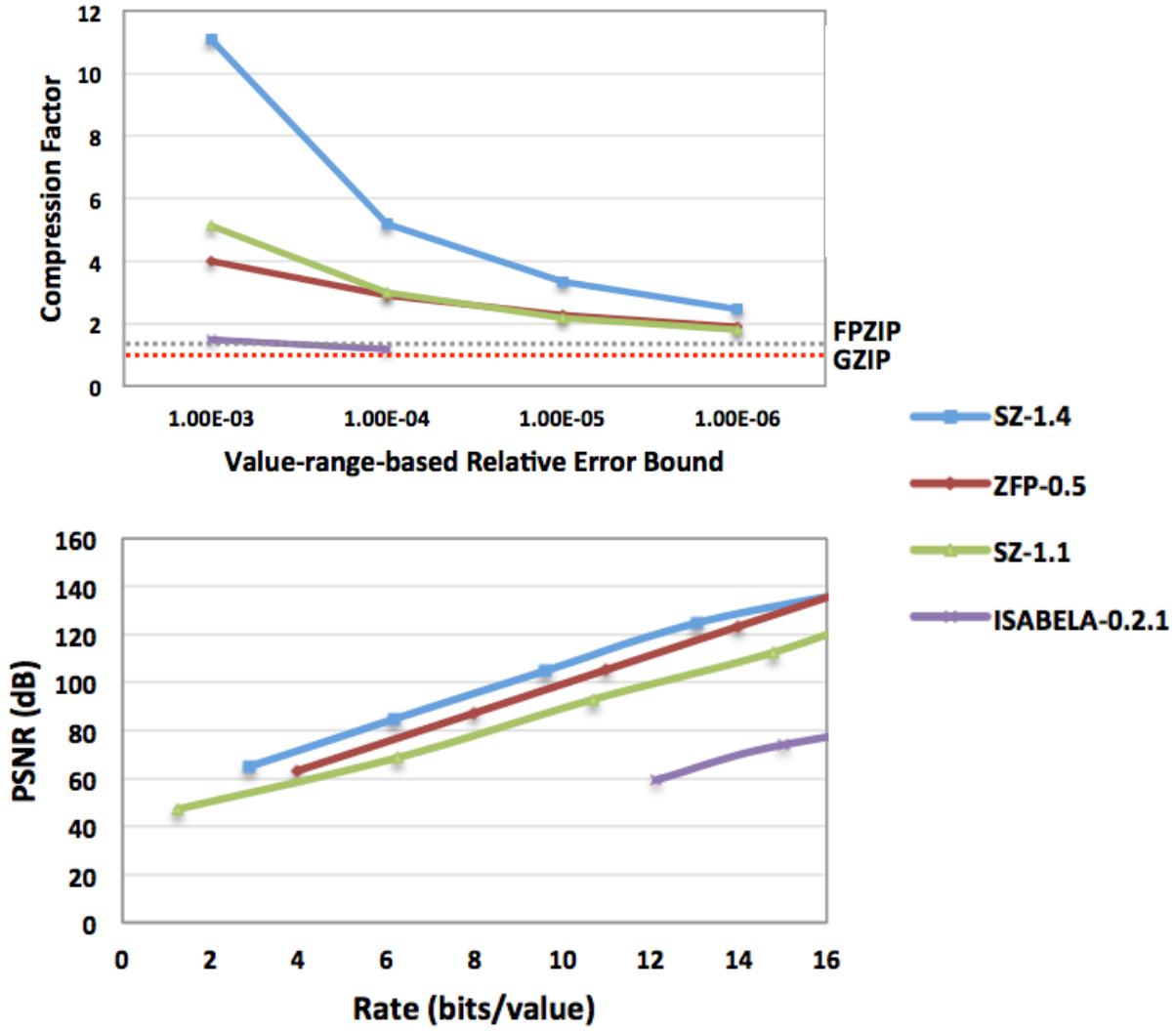
PSNR is logarithmic scale

Instrument Datasets

2D X-ray datasets from Argonne Photon Source (APS) instrument

- Data Source: X-ray instrument
- Dimension Size: 2560×2560
- Data Size: 40 GB
- File Number: 1518

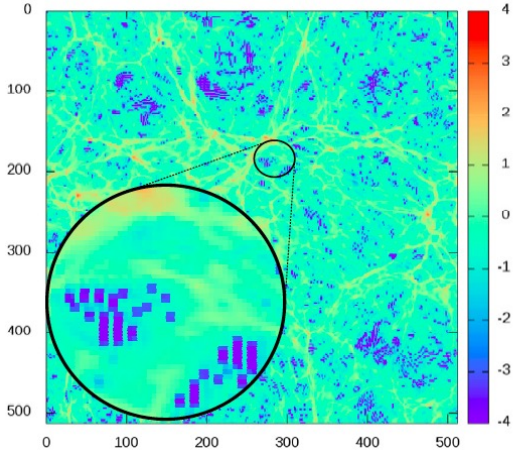




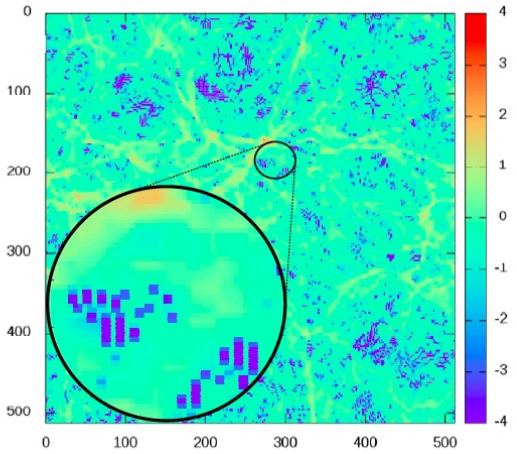
Visualization with SZ

LUDDY
SCHOOL OF INFORMATICS, COMPUTING, AND ENGINEERING

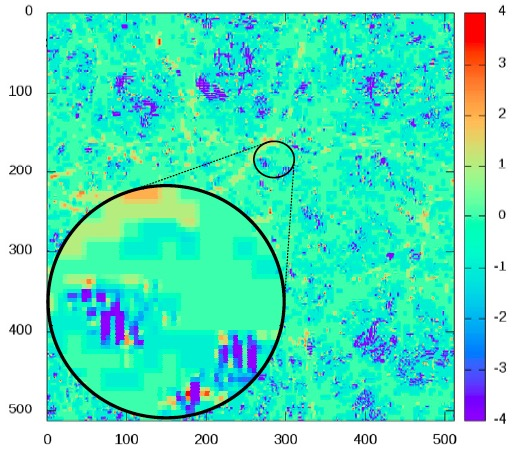
Cosmology (CR=**58:1**)



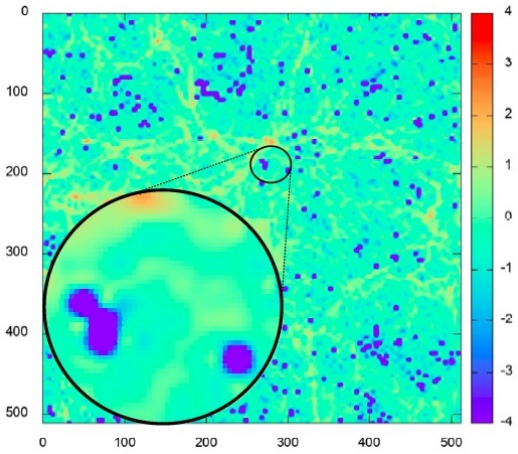
original raw data



SZ-2.0 (PSNR=**29**, SSIM=**0.6867**)

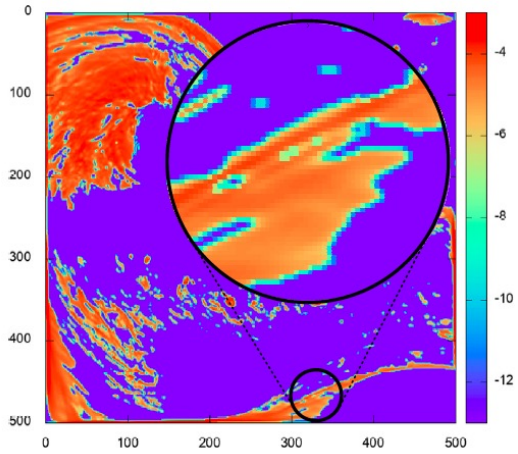


ZFP (PSNR=21.3, SSIM=0.3762)

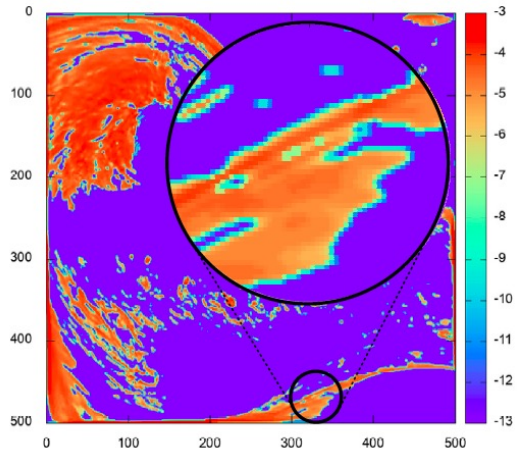


Downsamping + interpolation (PSNR=18.1, SSIM=0.4345)

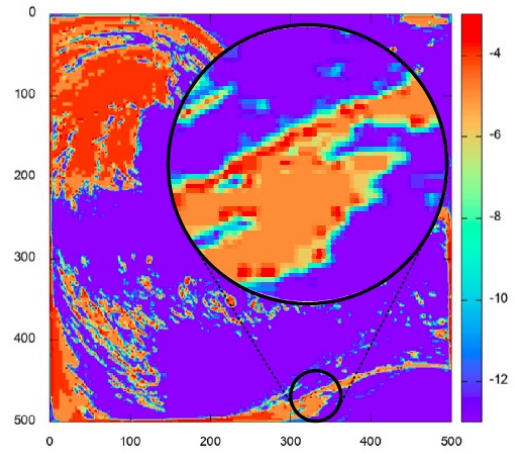
Hurricane (CR=**66:1**)



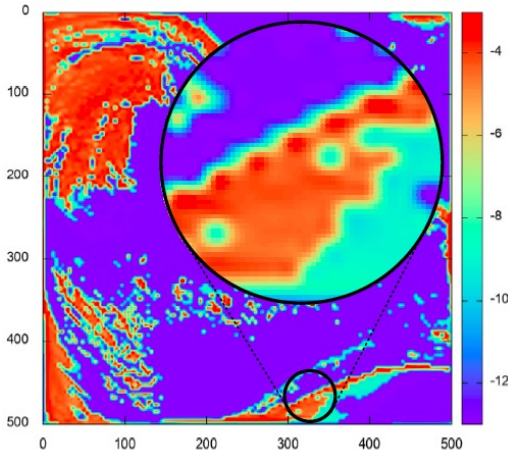
original raw data



SZ-2.0 (PSNR=**51**, SSIM=**0.9966**)



ZFP (PSNR=22.5, SSIM=0.8893)



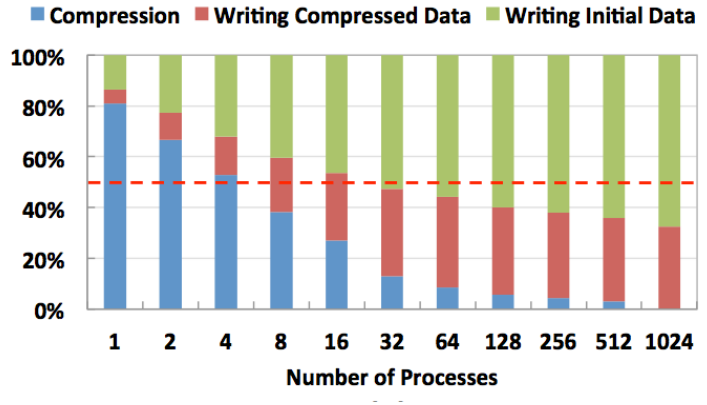
Downsamping + interpolation (PSNR=17.7, SSIM=0.7681)

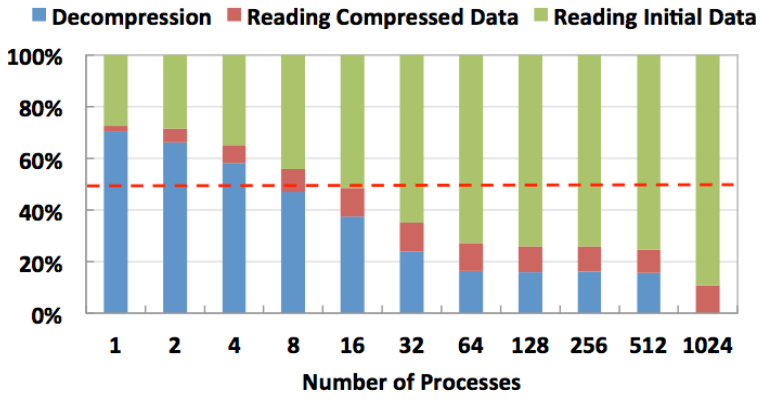
Parallel Evaluation

LUDDY
SCHOOL OF INFORMATICS,
COMPUTING, AND ENGINEERING

Climate

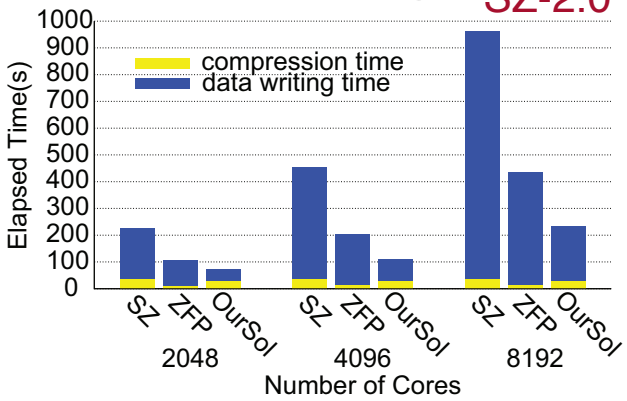
SZ-1.4



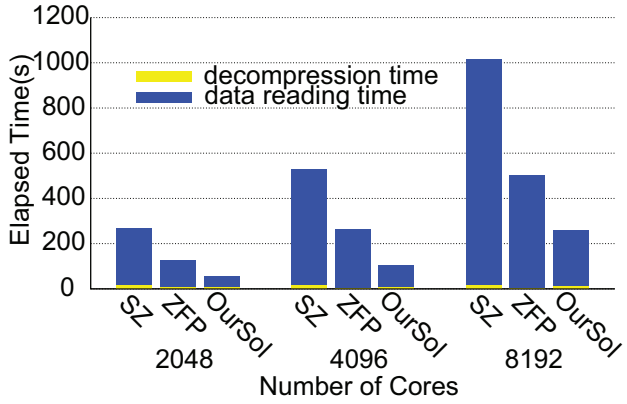


Cosmology

SZ-2.0



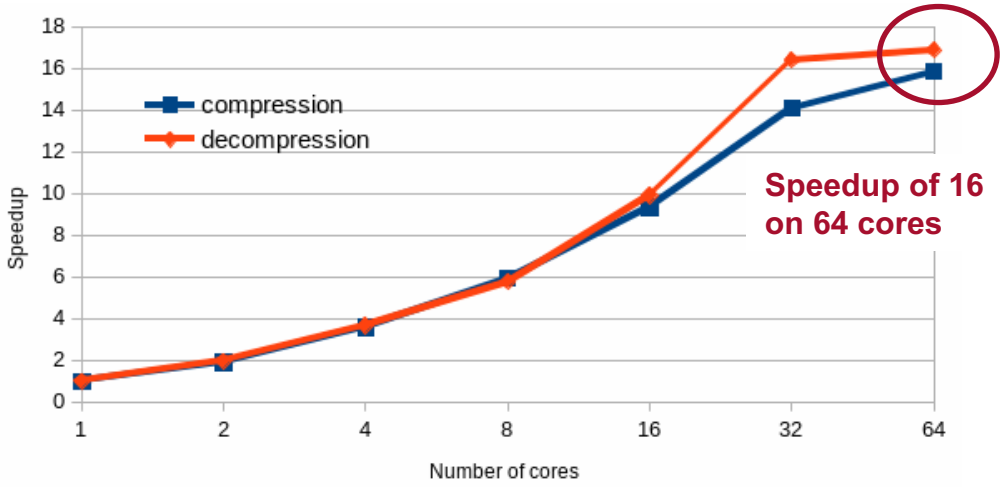
Data dumping time



Data loading time

Number of Processes	Number of Nodes	Comp Speed (GB/s)	Speedup	Parallel Efficiency
1	1	0.09	1.00	100.0%
2	2	0.18	2.00	99.8%
4	4	0.35	3.99	99.9%
8	8	0.70	7.99	99.8%
16	16	1.40	15.98	99.9%
32	32	2.79	31.91	99.7%
64	64	5.60	63.97	99.9%
128	64	11.2	127.6	99.7%
256	64	21.5	245.8	96.0%
512	64	40.5	463.0	90.4%
1024	64	81.3	930.7	90.9%

Speedup of 58 on 64 nodes



Speedup of 16 on 64 cores

cuSZ: An Efficient GPU Based Error-Bounded Lossy Compression Framework for Scientific Data

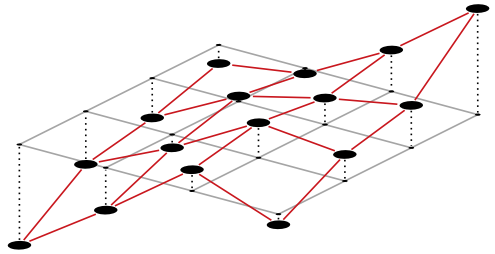
Published in *2020 International Conference on Parallel Architectures and Compilation Techniques (PACT'20)*

System Design

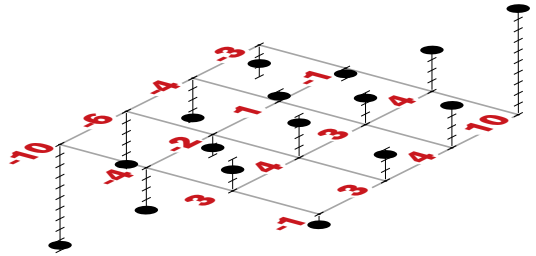
Challenges

- Tight data dependency—loop-carried *read-after-write* (RAW)—hinders parallelization.
- Host-device communications due to only considering CPU/GPU suitableness.

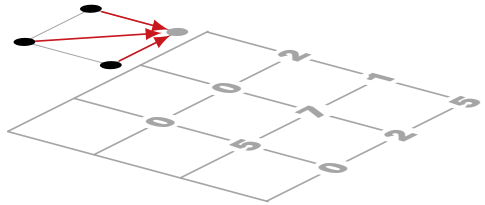
DUAL-QUANTIZATION AND PREDICTION



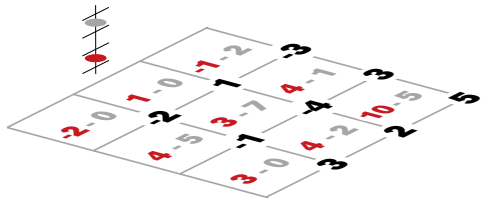
floating-point representation
original data



in units of *eb*
PREQUANTIZATION (NO RAW)



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

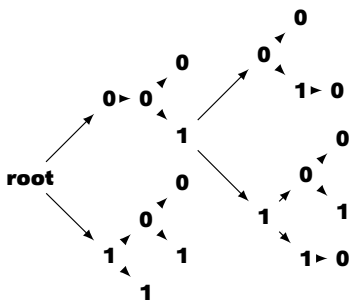


in units of *eb* (unchanged)
POSTQUANTIZATION (NO RAW)

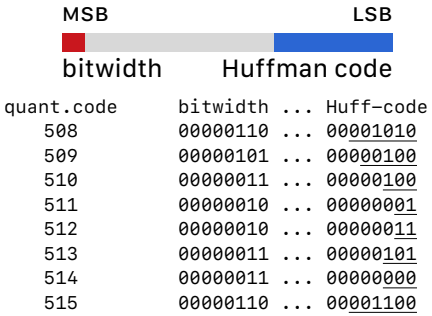
CUSTOMIZED HUFFMAN ENCODING

	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

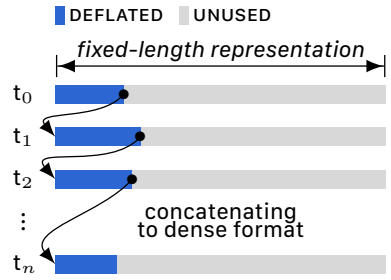


build and canonize Huffman codebook



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

Fully Parallelized P+Q

- 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:

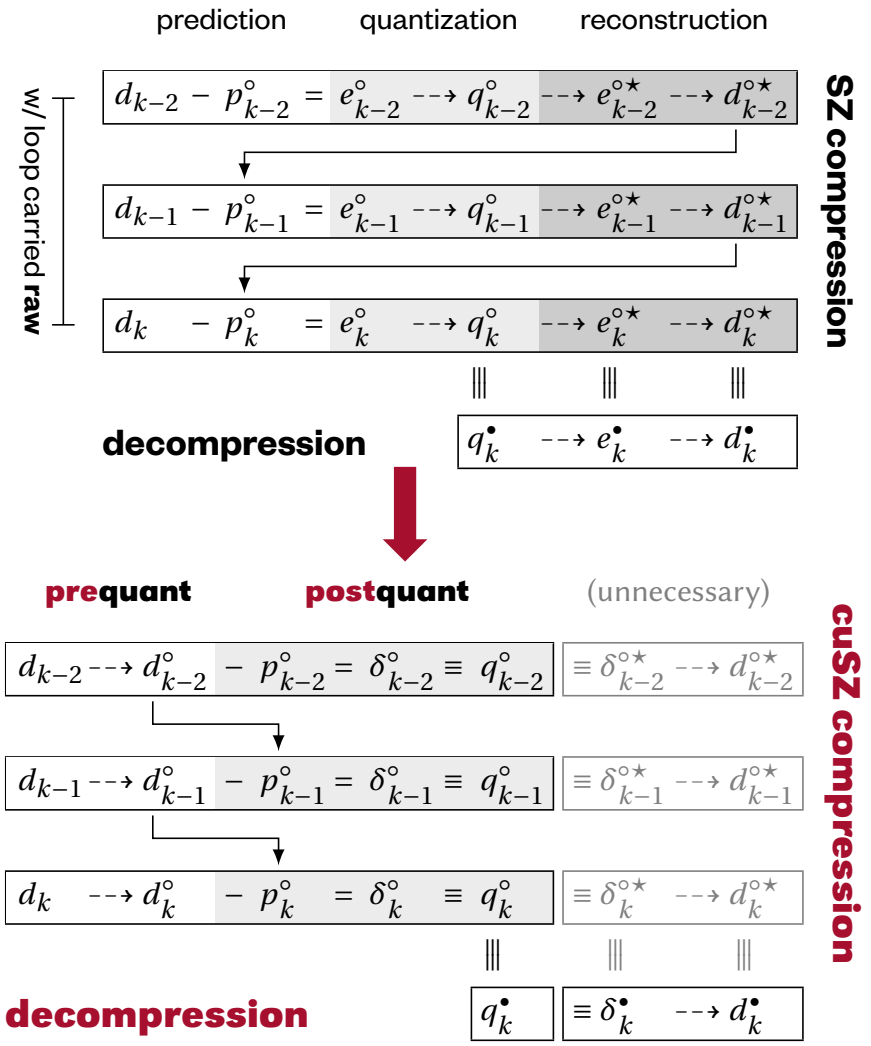
$$e^\circ/(2\cdot eb)\times(2\cdot eb)-e^\circ\le eb.$$

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

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

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

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



GPU Performance Optimization

Canonical Codebook & 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
  update: 0.8 ms
- canonize for 200 us (1024 symbols)
  update: incoporated in tree-building
- 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.

fine-grained manner:

IPDPS'21: *Revisiting Huffman Coding: Toward Extreme Performance on Modern GPU Architectures*, Tian et al.
IPDPS'22: *Optimizing Huffman Decoding for Error-Bounded Lossy Compression on GPUs*, Rivera et al.

	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.

Adaptive Parallelism

Worth noting: in canonizing codebook

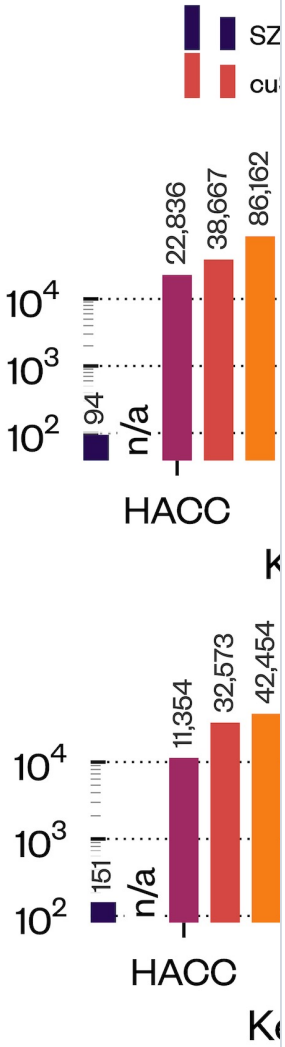
- problem size > max. `block` size (1024)
- utilize `cooperative groups` and `grid.sync()`
- `__syncthreads()`: not able
- `cudaDeviceSynchronize()`: expensive

Threads # Tuning

	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.

cuSZ –



cuSZ

A CUDA-Based Error-Bounded Lossy Compressor for Scientific Data

License BSD 3-Clause

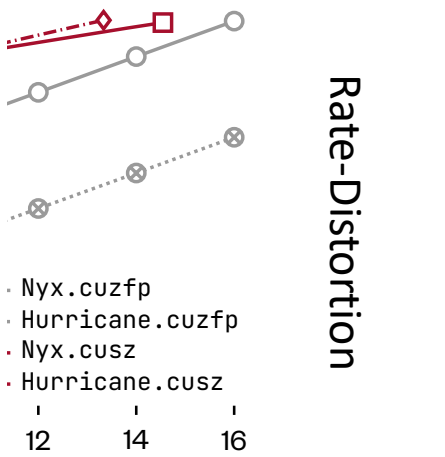
cuSZ is a CUDA implementation of the world-widely used SZ lossy compressor. It is the *first* error-bounded lossy compressor on GPU for scientific data, and it aims to improve SZ's throughput significantly on GPU-based heterogeneous HPC systems.

Our published papers cover the essential design and implementation.

- **PACT '20: cuSZ**, via local copy, via ACM, via arXiv
  - framework: (fine-grained) *N*-D prediction-based error-controling "construction" + (coarse-grained) lossless encoding
- **CLUSTER '21: cuSZ+**, via local, via IEEEXplore
  - optimization in throughput, featuring fine-grained *N*-D "reconstruction"
  - optimization in compression ratio, when data is deemed as "smooth"

Kindly note: If you mention cuSZ in your paper, please cite using these BibTeX entries.

- developers: Jiannan Tian, Cody Rivera, Wenyu Gai, Dingwen Tao, Sheng Di, Franck Cappello
- contributors (alphabetic): Jon Calhoun, Megan Hickman Fulp, Xin Liang, Robert Underwood, Kai Zhao
- Special thanks to Dominique LaSalle (NVIDIA) for serving as Mentor in Argonne GPU Hackaton 2021!



Accelerating Parallel Write via Deeply Integrating Predictive Lossy Compression with HDF5

To appear in *International Conference for High Performance Computing, Networking, Storage, and Analysis (ACM/IEEE SC'22)*

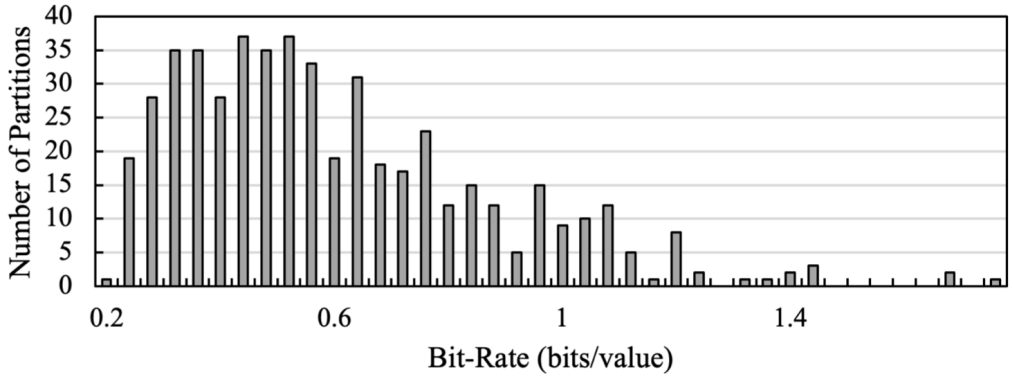
Introduction

Current Limitations

- **Sequential** compression and I/O
- Offset **cannot** be simply pre-assigned
  - Compression sizes vary drastically across different data partitions

Our Solution and Contributions

- Integrate predictive lossy compression (such as SZ) with asynchronous I/O
- Extend prediction model to **estimate** the offset and time of parallel I/O
- **Overlap** I/O with compression
- **Optimize** order of compression tasks to achieve higher performance



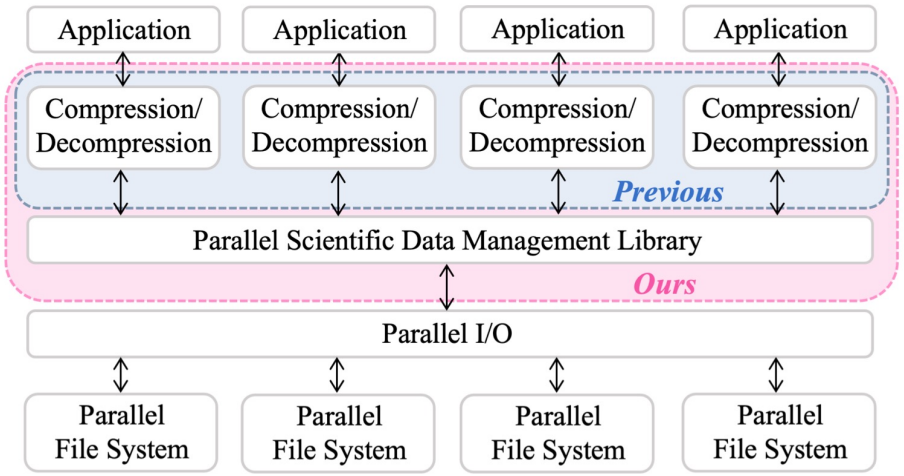
↑ Compression bit-rate distribution on a Nyx dataset with 512 partitions. Every partition uses the same compression configuration.

- Our solution improves the HDF5 parallel-write performance by up to **4.5**× and **2.9**× compared to two existing solutions: parallel write (1) without compression and (2) with the SZ lossy compression filter, respectively, with only 1.5% storage overhead

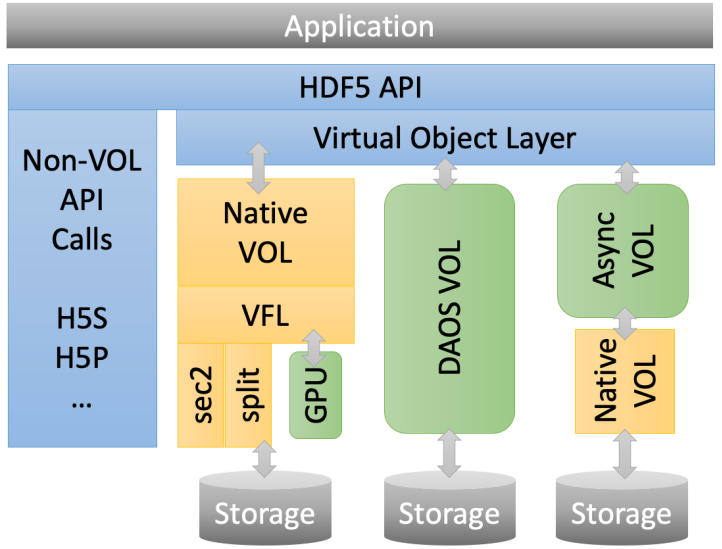
Background

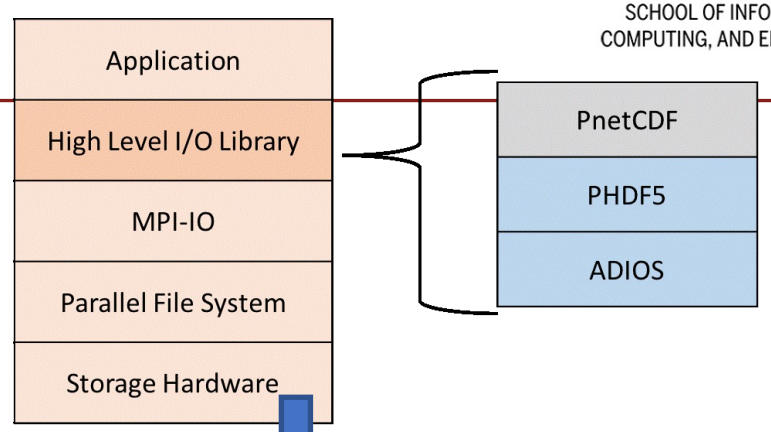
Parallel I/O Libraries for HPC Applications

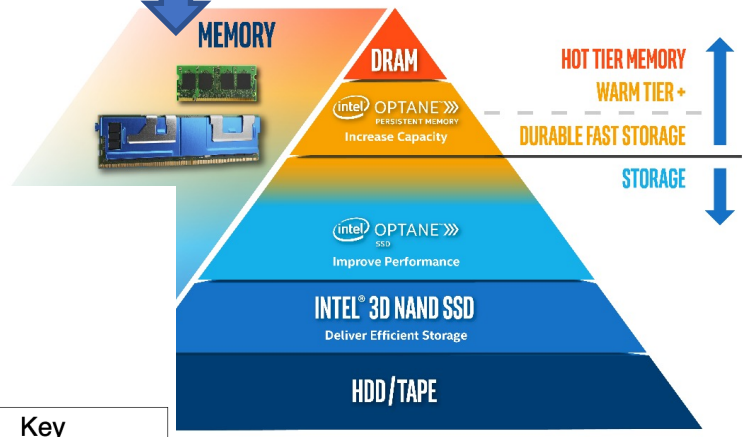
- Access and manage scientific data efficiently
- Moving data between compute nodes and complex storage
  - Node-local persistent memory, burst buffers, disk-based storage, etc.
- Currently compression is a dedicated layer (e.g., HDF5's dynamically loaded filter) in between applications and I/O libraries
  - E.g., HDF5 filter based on SZ: https://github.com/disheng222/H5Z-SZ
- HDF5 virtual object layer (VOL): redirect I/O operations into VOL connector and allow asynchronous I/O https://www.hdfgroup.org/wp-content/uploads/2020/10/Virtual-Object-Layer-VOL-Intro.pdf

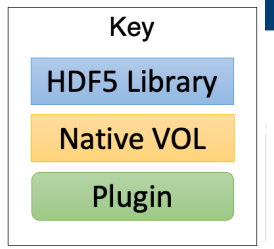


↑ Scientific data management with compression.









HDF5 Ecosystem

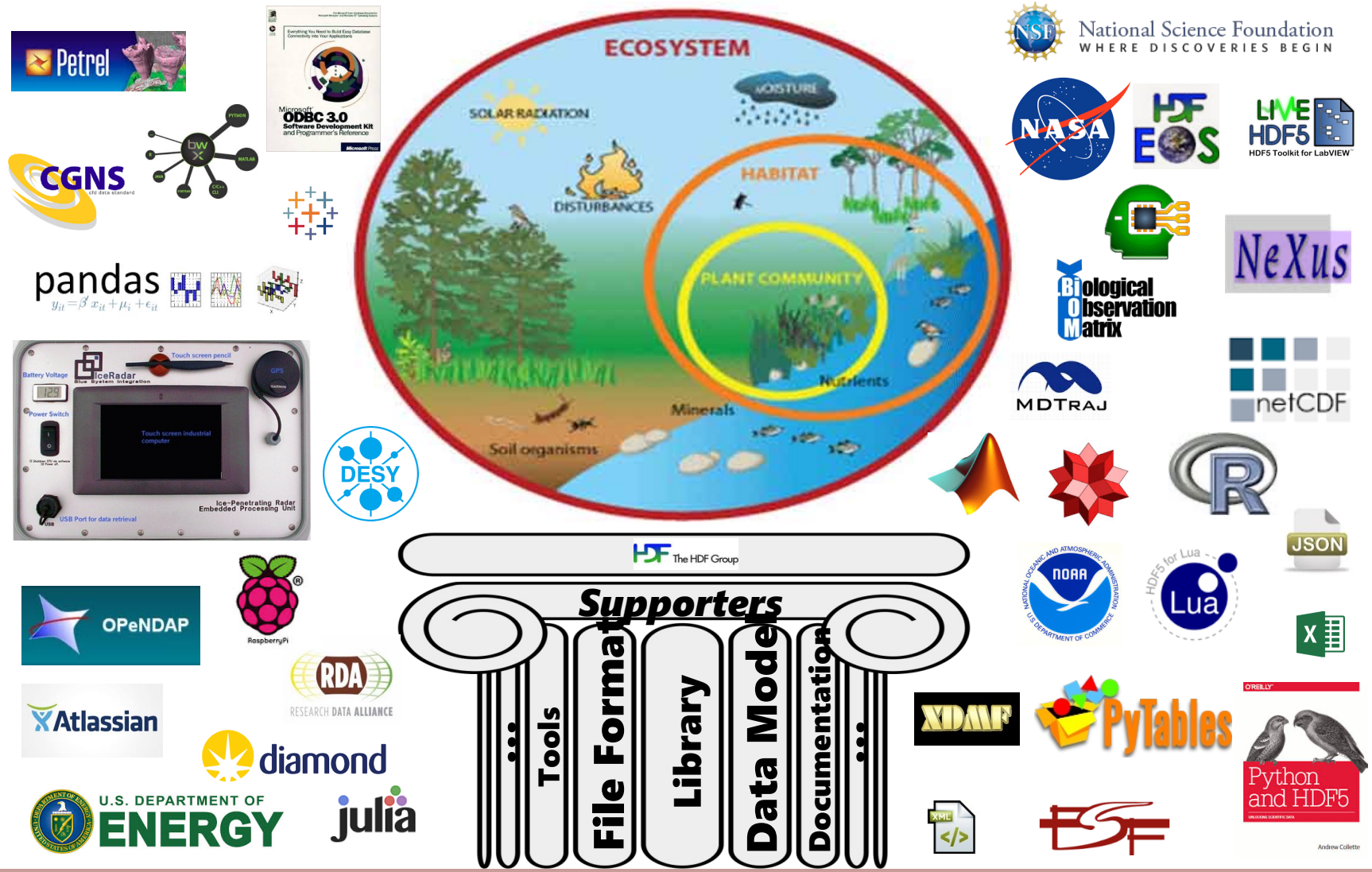


Figure from Q. Koziol (LBL)

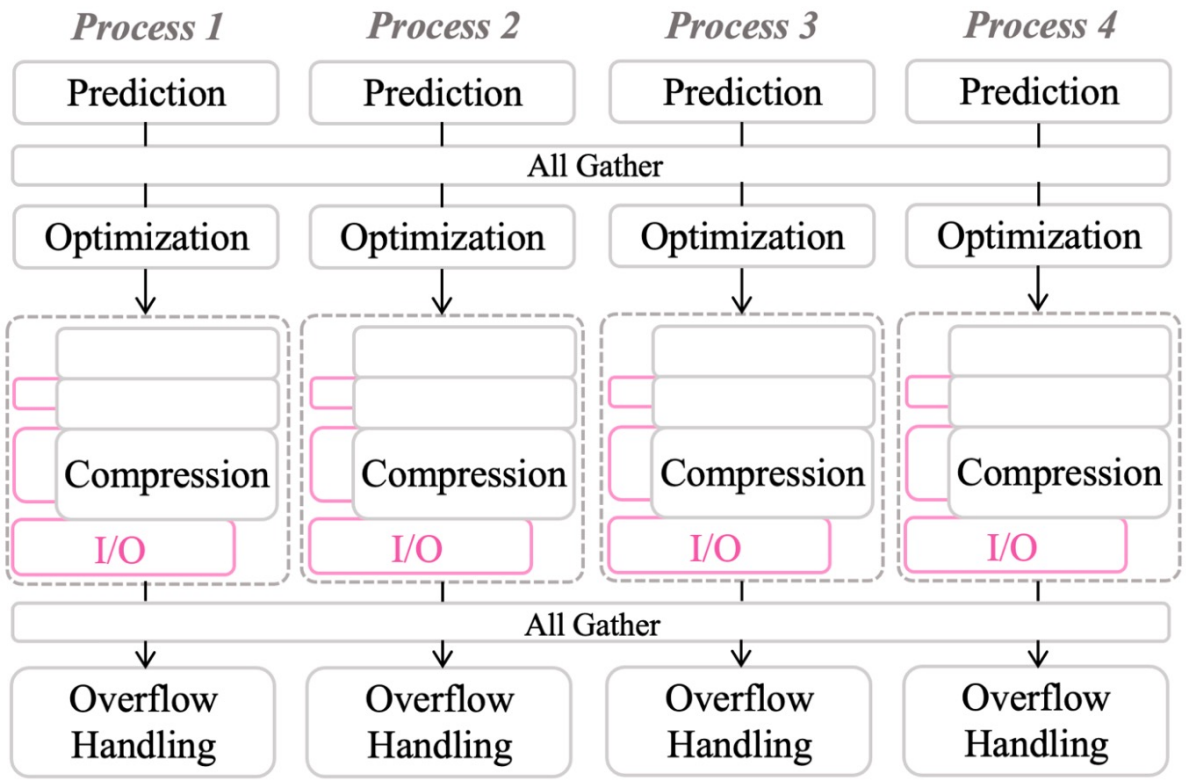
Example of a PHDF5 C Program

A parallel HDF5 program has a few extra calls

```
MPI_Init(&argc, &argv);

fapl_id = H5Pcreate(H5P_FILE_ACCESS);
H5Pset_fapl_mpio(fapl_id, comm, info);
file_id = H5Fcreate(FNAME,…, fapl_id);
space_id = H5Screate_simple(…);
dset_id = H5Dcreate(file_id, DNAME, H5T_NATIVE_INT,
                    space_id,…);
xf_id = H5Pcreate(H5P_DATASET_XFER);
H5Pset_dxpl_mpio(xf_id, H5FD_MPIO_COLLECTIVE);
status = H5Dwrite(dset_id, H5T_NATIVE_INT, …, xf_id…);
MPI_Finalize();
```

Design Methodology

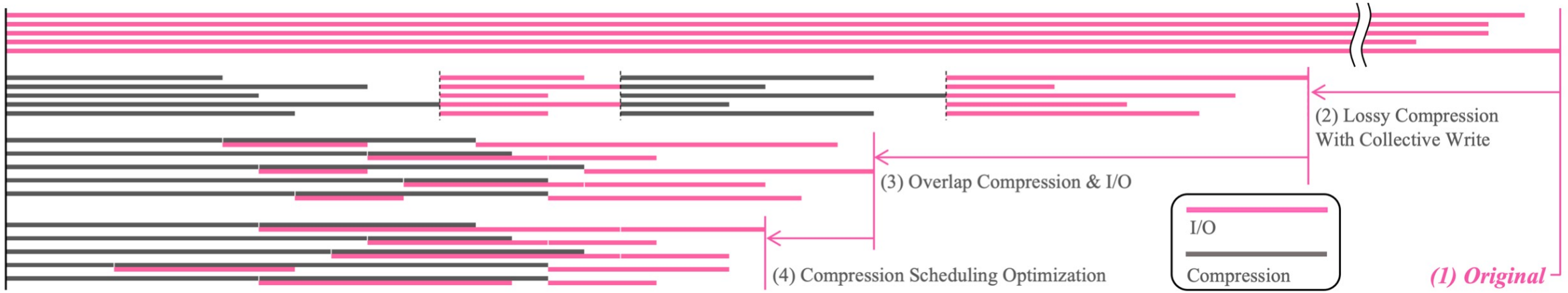


↑ Overview of our proposed solution.

Overall Design

1. **Predict** compression ratio and throughput
2. Distribute estimated compression ratio of each partition to all processes
3. Compute **offset** (compressed size) for parallel write
4. **Optimize** the order of compressing different data fields in each process
5. **Overlap** compressions and writes
6. Distribute overflow information
7. Handle **overflowed** data

Design Methodology



↑ Timeline of data aggregation with 5 processes and 2 data fields.

How Our Solution Compares to Existing Solutions

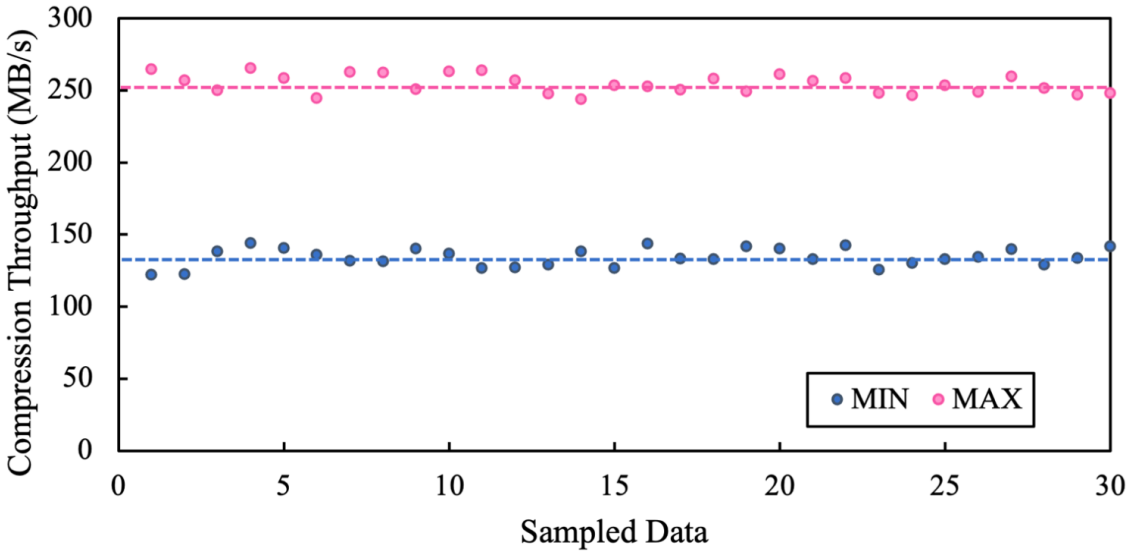
- Existing solutions:
  - (1) Original: non-compression solution
  - (2) Lossy compression solution using HDF5 filter
- Our Solutions:
  - (3) Overlap compression & I/O
  - (4) Overlap compression & I/O + compression scheduling optimization

Design Methodology

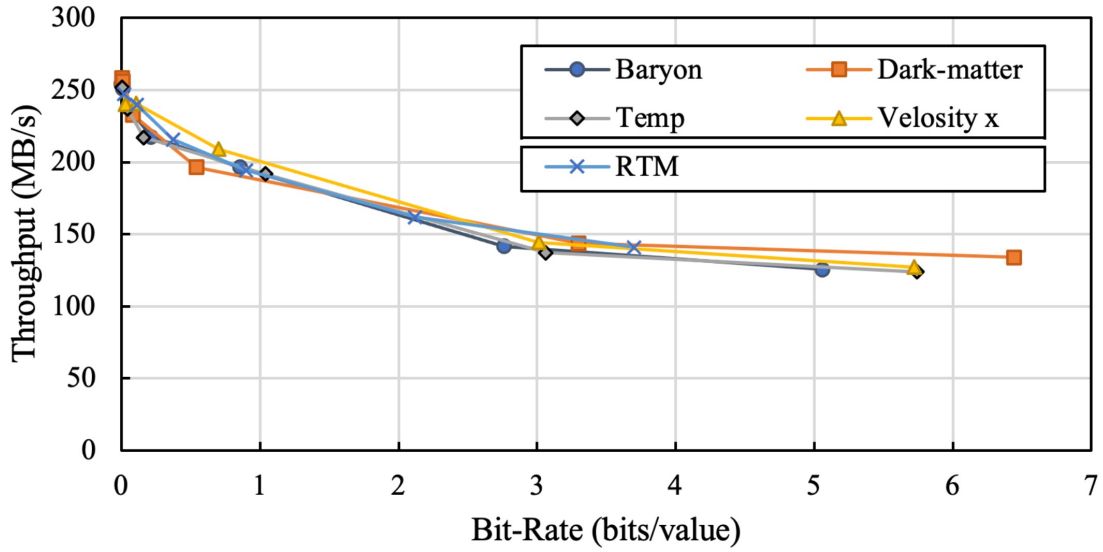
Compressor Throughput Estimation

- Min and max compression throughputs are similarly bounded across different data samples
- Bitrate-throughput curve for each data sample is highly consistent

$$T_{comp} = D/S$$
$$= (B_{ori} \times n)/(((C_{max} - C_{min}) \times 3^{-a})B^a + C_{min})$$



↑ Minimum and maximum compression throughput of a given data partition based on 30 samples from Baryon density



↑ Single-core compression throughput with different bit-rates on Nyx and RTM datasets

Design Methodology

Compressor Throughput Estimation

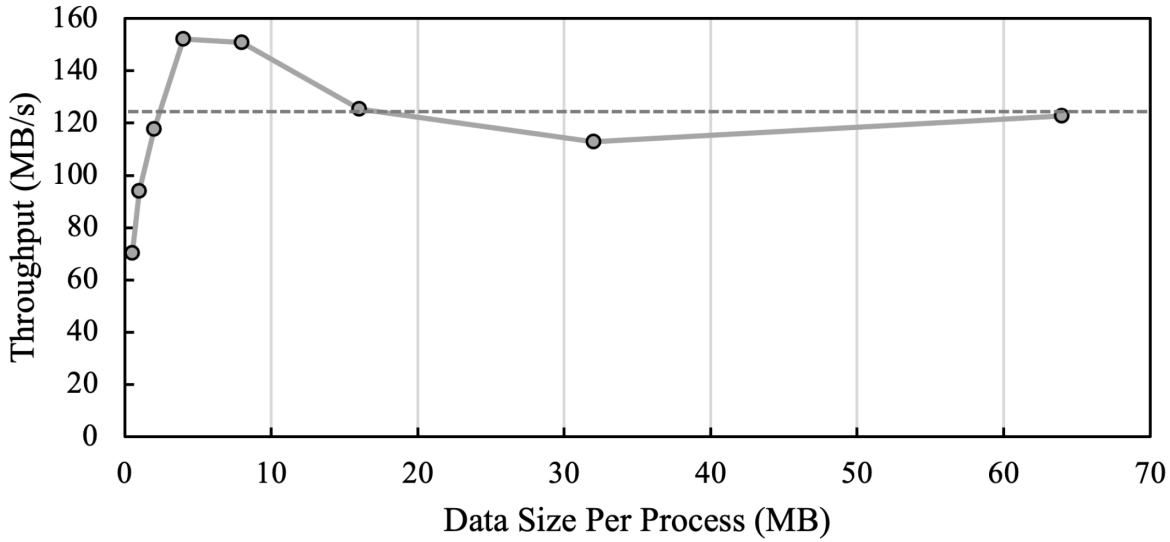
- Min and max compression throughputs are similarly bounded across different data samples
- Bitrate-throughput curve for each data sample is highly consistent

$$T_{comp} = D/S$$
$$= (B_{ori} \times n)/(((C_{max} - C_{min}) \times 3^{-a})B^a + C_{min})$$

Write Time Estimation

- Not to provide a highly accurate write-time estimation for each data partition, but to provide a capability to estimate the relative write time across different data sizes
- Write time stabilizes after data size reaches a certain point

$$T_{write} = (B \times n)/C_{thr}$$



↑ Independent write I/O throughput per process with different data sizes per process

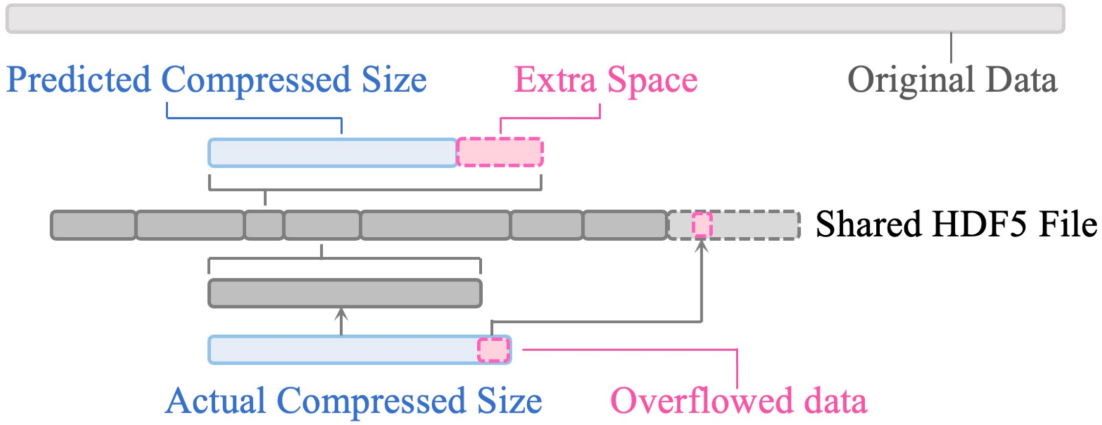
Design Methodology

Jin, S., Di, S., Jiannan, T., Byna, S., Tao, D. and Cappello, F., 2022, May. Improving Prediction-Based Lossy Compression Dramatically via Ratio-Quality Modeling. In Proceedings of *The 38th IEEE International Conference on Data Engineering (ICDE'22)*.

LUDDY SCHOOL OF INFORMATICS, COMPUTING, AND ENGINEERING

Overlapping Compression and Write

- Estimate/predict the **offset** (i.e., compressed data size) based on our previously built theoretical model
- Reserve an **extra space** for compressed data overflow
- Extra space ratio can be adjusted to balance between performance and compressed size overhead

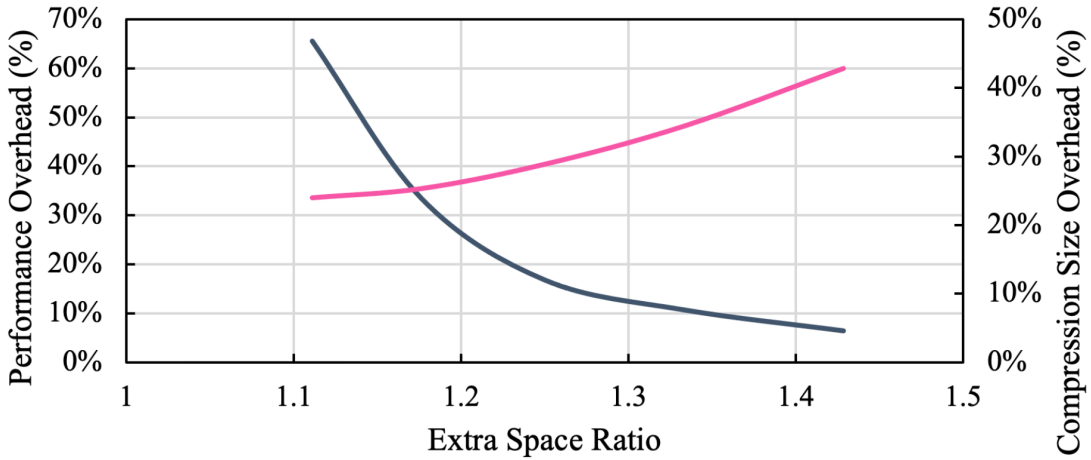


↑ Overflow data handling with preserved extra space.

Extra Space Ratio

- Default at 1.25 for most partitions
- Adjust for partitions with low estimated compression ratio

$$r_{space} = \min(2, 1 + (R_{space} - 1) \times 4),$$
$$where \quad r_{comp} > 32.$$

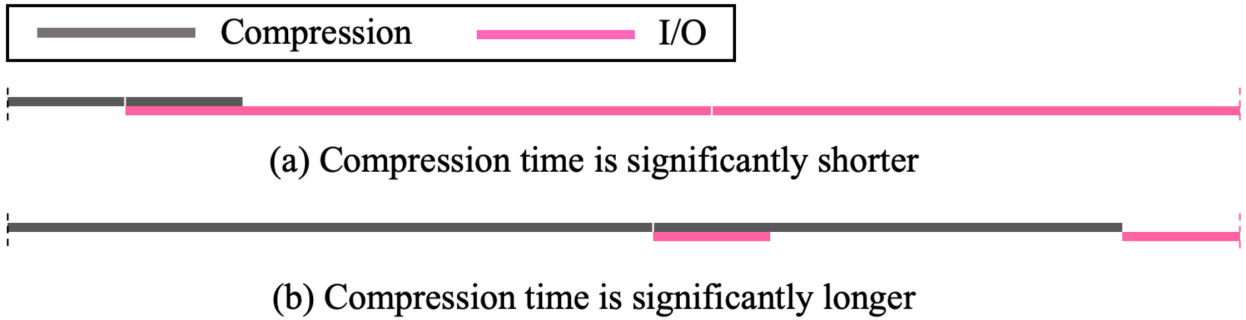


Trade-off between performance overhead and compression size overhead ↑

Design Methodology

Compression Order Optimization

- Improve overlapping efficiency
  - I/O of each partition happens after compression
  - Avoid unnecessary wait time for I/O
- When good? Compression time and I/O time are similar
- When not good?
  - I/O is significantly longer
  - Compression is significantly longer



(a) Compression time is significantly shorter

(b) Compression time is significantly longer

Algorithm 1 Compression Order Optimization

Notation: data fields in current process: ℓ; compression queue: Q; compression queue after insert and additional data: Q°; possible insert locations in a queue: β; time to compress: t_c; time to write:t_w; predicted compression time: $P_c(\ell)$; predicted write time: $P_w(\ell)$

Global: $P_c(\ell), P_w(\ell)$

```
1  procedure TIME(q)
2      t_c, t_w ← 0
3      for ℓ ← data fields in q do
4          t_c ← t_c + P_c(ℓ)
5          t_w ← P_w(ℓ) + max(t_c, t_w)
6      end for
7      return t_w
8  end procedure
9
10 procedure SCHEDULINGOPTIMIZATOR
11     for ℓ ← data fields in current process do
12         for β ← all possible insert location do
13             Q° ← insert ℓ to β
14             if TIME(Q°) < TIME(Q) or first β then
15                 Q ← Q°
16             end if
17         end for
18     end for
19     return Q
20 end procedure
```

↑ An example of extremely unbalanced compression time and write time, limiting the benefit from our reordering.

Evaluation

LUDDY
SCHOOL OF INFORMATICS, COMPUTING, AND ENGINEERING

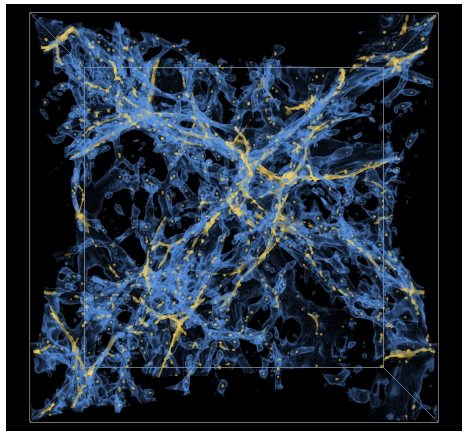
Experimental Setup

- Implemented our approach with HDF5 and SZ3
- Two HPC systems
  - **Summit supercomputer** at Oak Ridge National Lab
  - **Bebop cluster** at Argonne National Lab
- Different scales of Nyx and VPIC datasets
  - Use PSNR to validate the reconstructed data quality
  - Both datasets result in ~16X compression ratio

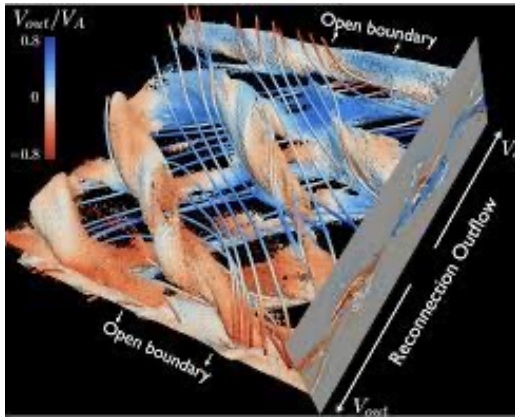
I/O-Intensive HPC Applications

Name	Description	Scale	Size
nyx [18]	Cosmology simulation	4096×4096×4096	2.47 TB
		2048×2048×2048	206.15 GB
		1024×1024×1024	25.76 GB
		512×512×512	3.22 GB
VPIC [52]	Particle simulation	161,297,451,573	4.62 TB

↑ Details of Tested Datasets.



Nyx cosmology simulation



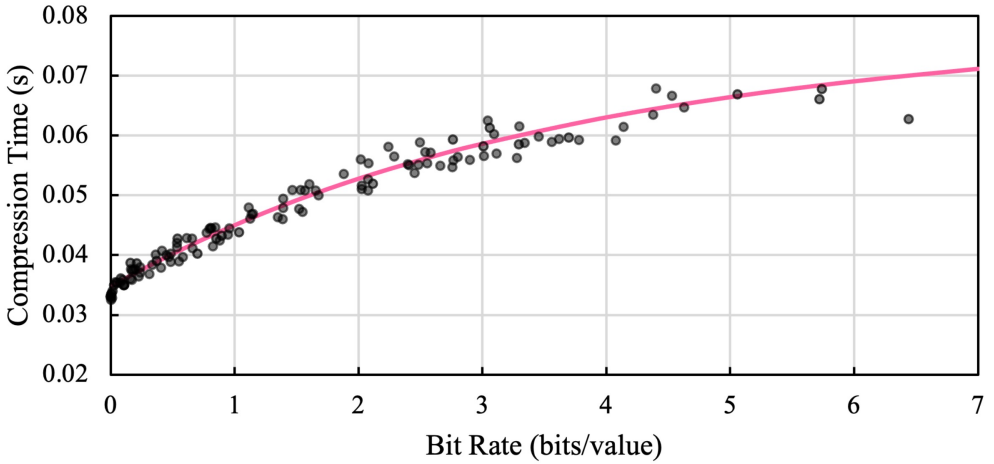
VPIC plasma simulation

Evaluation

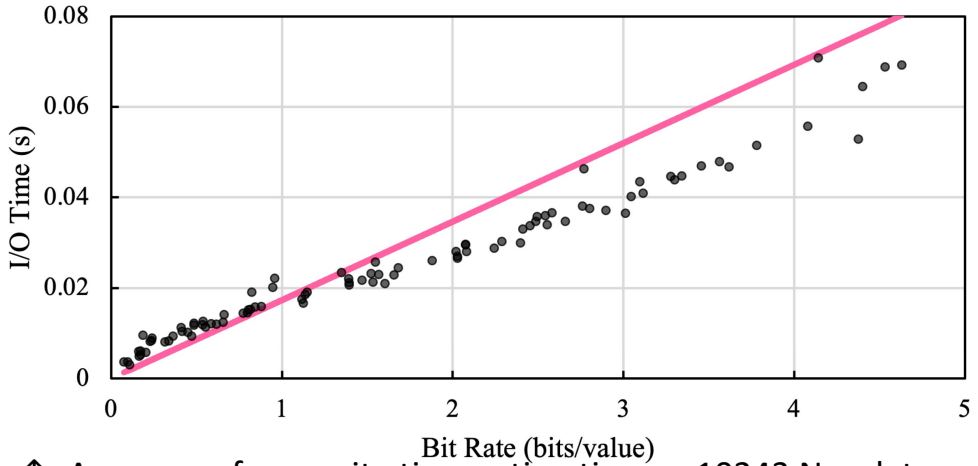
LUDDY
SCHOOL OF INFORMATICS, COMPUTING, AND ENGINEERING

Compression & I/O Throughput Estimation Accuracy

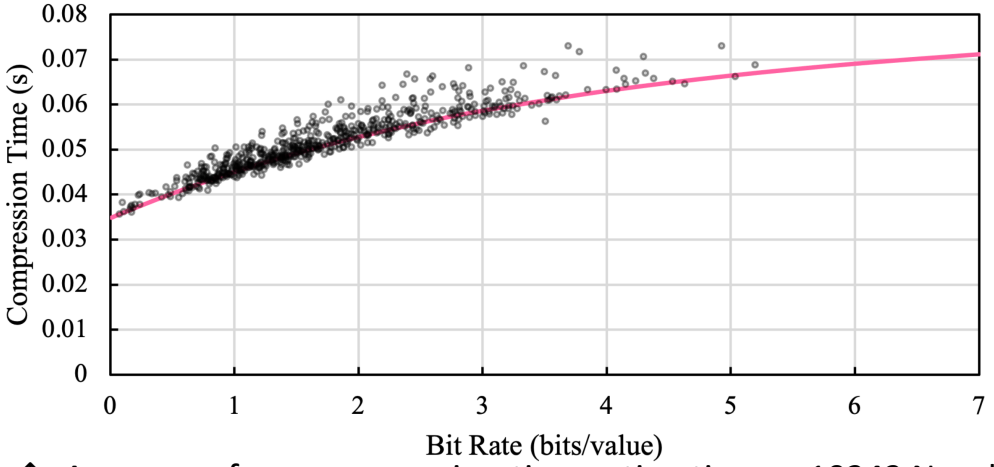
- High accuracy on compression time estimation
  - Different partitions
  - Different data scales
- High accuracy on write time estimation
  - Have some distortion but NOT affect our optimization



↑ Accuracy of our compression-time estimation on 5123 Nyx data samples (red line is predicted time; black dots are actual time)

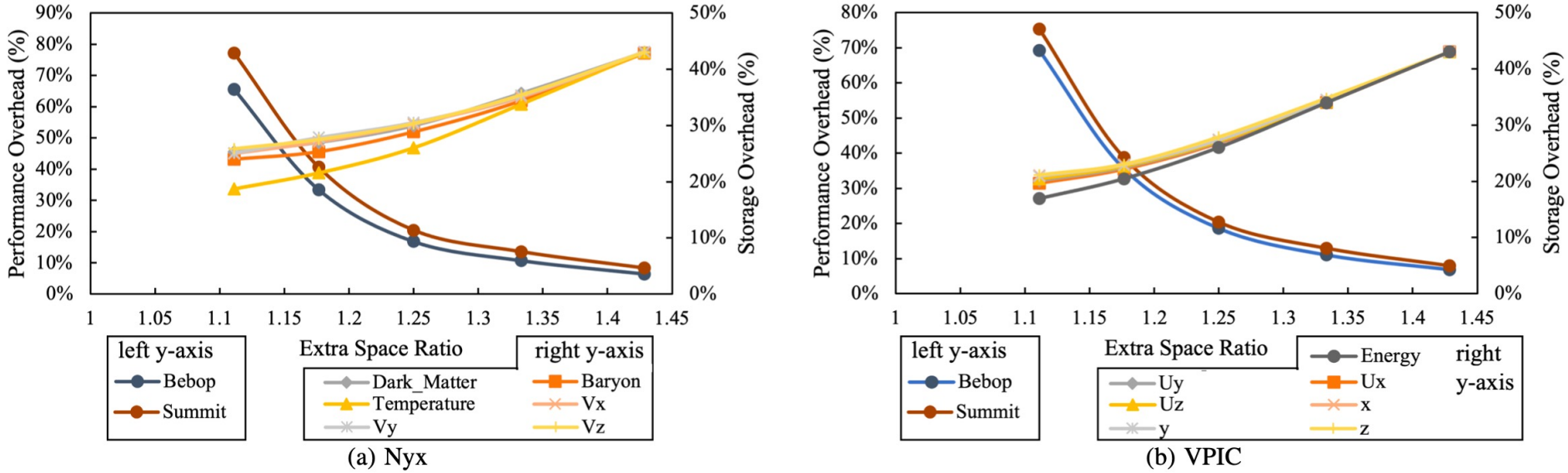


↑ Accuracy of our write time estimation on 10243 Nyx data samples. Red line is predicted time; black dots are actual time.



↑ Accuracy of our compression-time estimation on 10243 Nyx data samples. Red line is predicted time; black dots are actual time.

Evaluation



↑ Trade-off between performance overhead and storage overhead based on different extra space ratios on Nyx dataset (6 data fields) and VPIC dataset (7 data fields) on both Bebop and Summit with 512 processes.

Evaluation on Extra Space Ratio

- Trade-off curve between performance and storage are highly similar
- Lower the extra space ratio can result in extremely high performance overhead
- We can use **the same extra space ratio** for different setups (default at 1.25)
- Users can also custom the extra space ratio

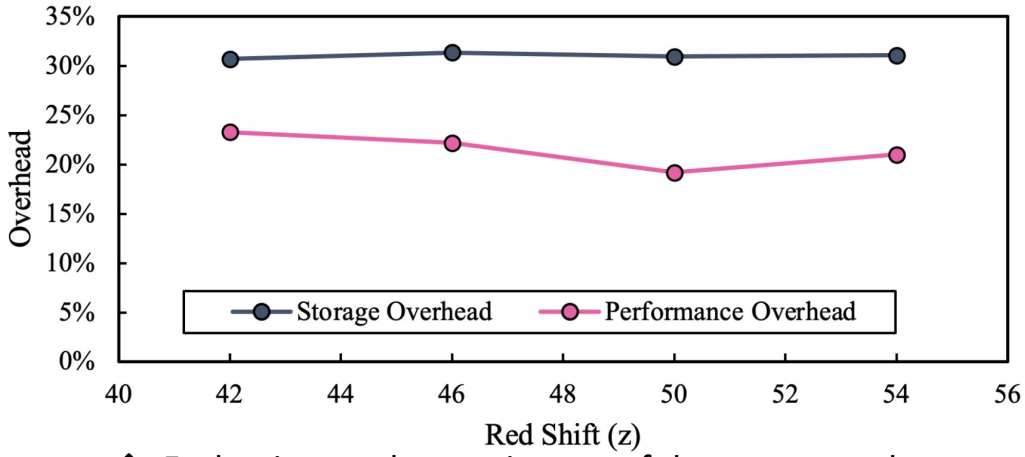
Evaluation

Comparison

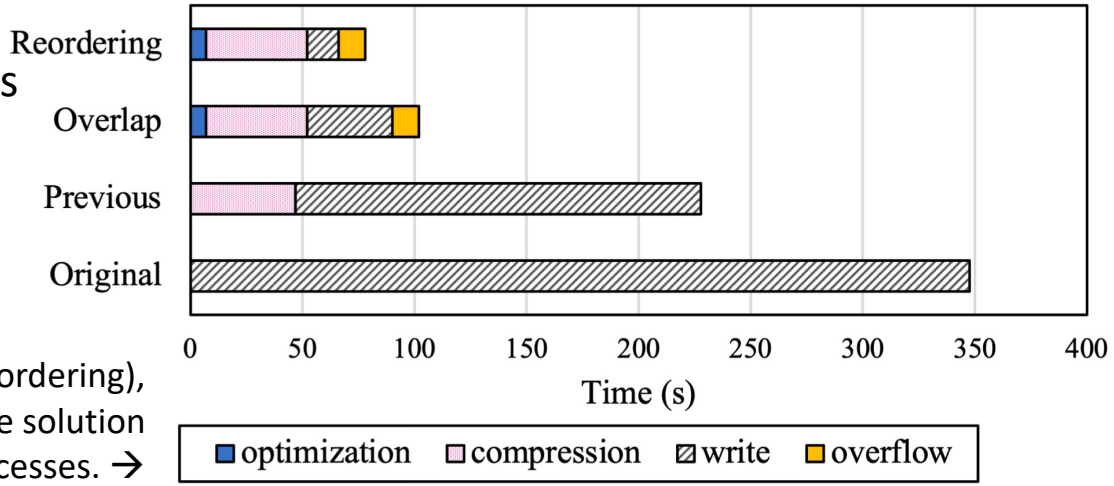
- Original: non-compression solution
- Previous: compression filter solution
- Overlap: our solution
- Reordering: overlap + reorder technique

Performance Improvement

- Original → Previous: 1.87×
- Previous → Overlap: 1.79×
- Overlap → Reordering: 1.30×
- Overall: **2.91×** improvement from previous with a 26% storage overhead. 1.5% if compared to original size
- **Stable** performance over timesteps

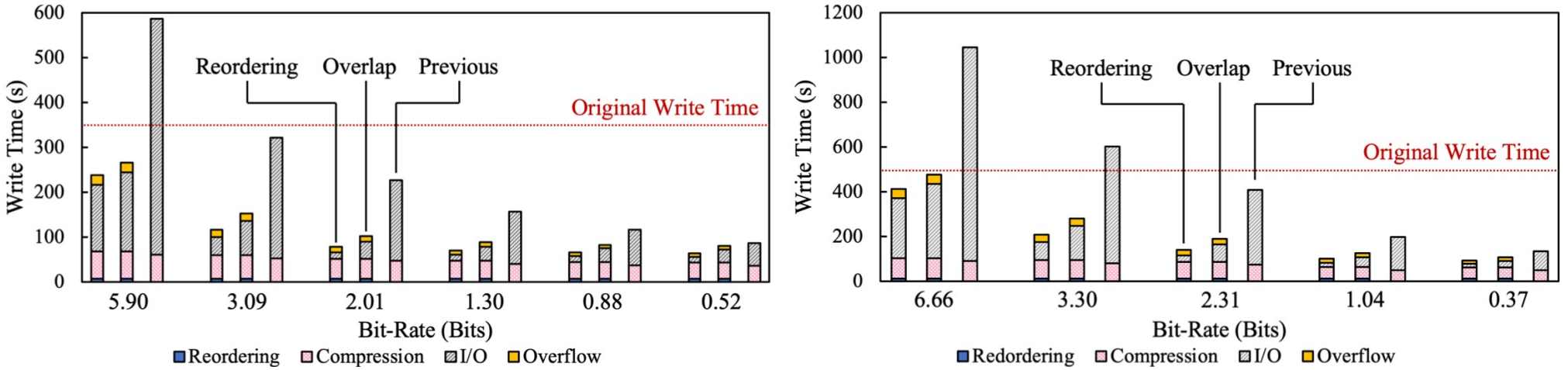


↑ Evaluation on the consistency of the storage and performance overheads using the same extra space ratio of 1.25 with 512 processes on Summit.



Performance comparison among our solution (overlapping and reordering), original non-compression solution, and previous compression-write solution on 4096^3 Nyx dataset with 512 processes. →

Evaluation

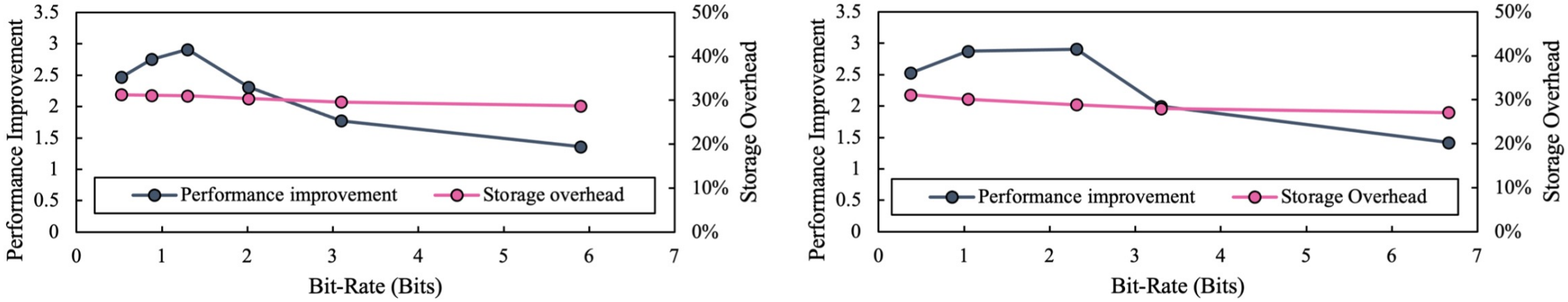


↑ Performance improvement of overall parallel-write with our proposed solution compared to the previous write solution with H5Z-SZ on both Nyx and VPIC datasets. Dashed red line is the baseline of HDF5 without compression. (a) and (b) are evaluated with 512 processes on Summit.

Performance with Different Overall Ratios

- Limited improvement from reordering optimization under extremely high/low bit-rate
  - High bit-rate: I/O time significantly larger than compression time
  - Low bit-rate: compression time significantly larger than I/O time
- Storage overhead is **stable** (~20% of compressed data)
- Performance improvement is more significant with **higher bit-rate**
  - Low bit-rate: compression time dominate, little overlap efficiency

Evaluation



(a) Nyx with different compression ratio

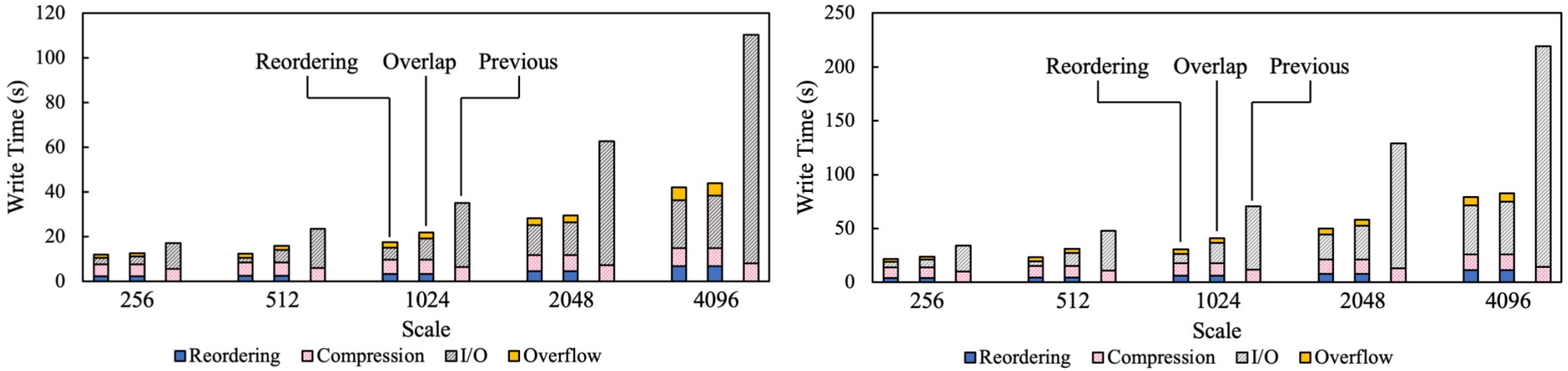
(b) VPIC with different compression ratio

↑ Performance improvement (overall) and storage overhead of our solution compared to the previous solution on both Nyx and VPIC datasets. (a) and (b) are evaluated with 512 processes om Summit.

Performance with Different Overall Ratios

- Limited improvement from reordering optimization under extremely high/low bit-rate
  - High bit-rate: I/O time significantly larger than compression time
  - Low bit-rate: compression time significantly larger than I/O time
- Storage overhead is **stable** (~20% of compressed data)
- Performance improvement is more significant with **higher bit-rate**
  - Low bit-rate: compression time dominate, little overlap efficiency

Evaluation

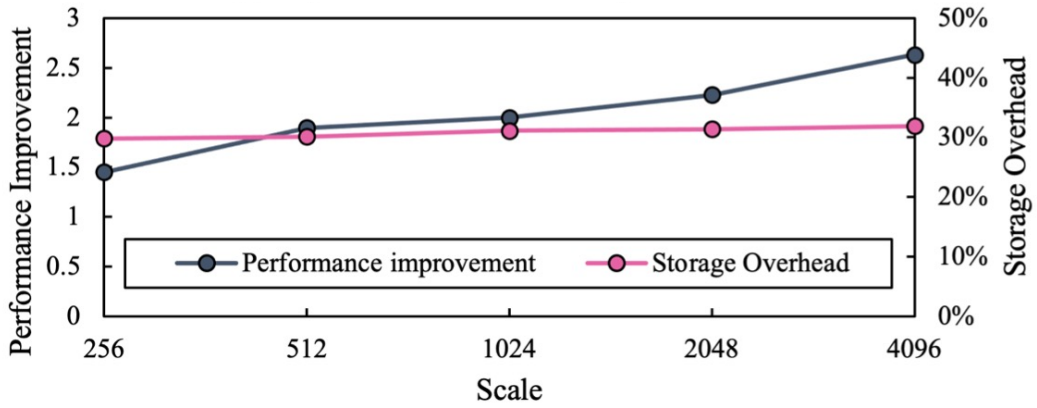


↑ Performance improvement of overall parallel-write with our proposed solution compared to the previous write solution with H5Z-SZ on both Nyx and VPIC datasets. Dashed red line is the baseline of HDF5 without compression. (c) and (d) are evaluated with a target bit-rate of 2.

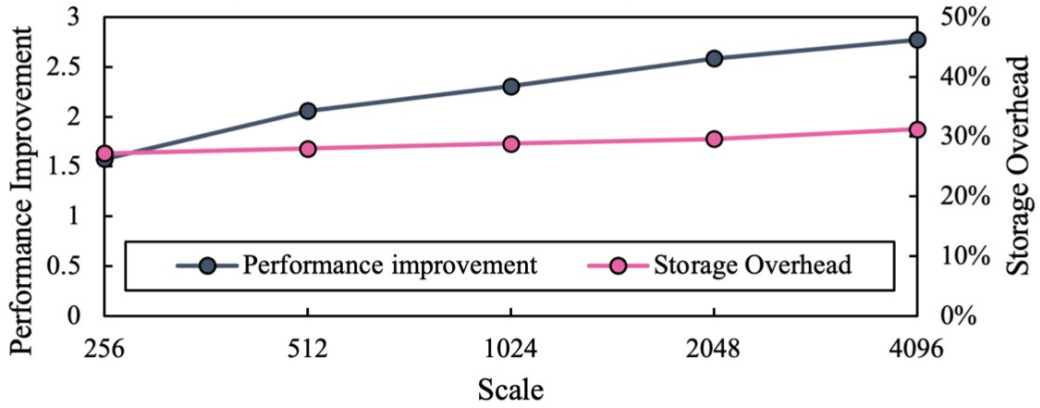
Performance with Different Scales

- Improvement from reordering optimization is **stable** (~22%)
- Storage overhead is **stable** (~20% of compressed data)
- Performance improvement is more significant with **larger scale**
  - Independent write provides better scalability than collective write (used by previous comp-write solution)

Evaluation



(c) Nyx with different scale



(d) VPIC with different scale

↑ Performance improvement (overall) and storage overhead of our solution compared to the previous solution on both Nyx and VPIC datasets. (c) and (d) are evaluated with a target bit-rate of 2.

Performance with Different Scales

- Improvement from reordering optimization is **stable** (~22%)
- Storage overhead is **stable** (~20% of compressed data)
- Performance improvement is more significant with **larger scale**
  - Independent write provides better scalability than collective write (used by previous comp-write solution)

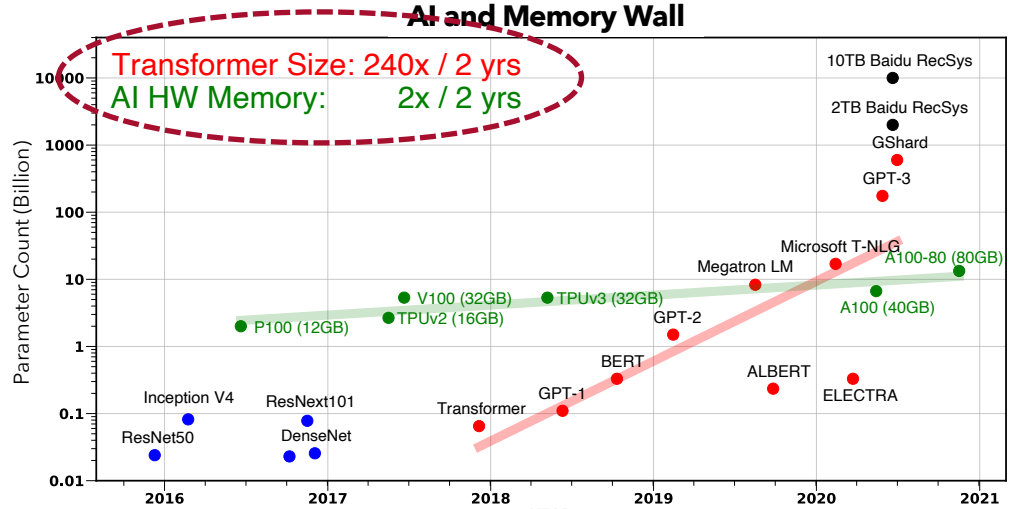
COMET: A Novel Memory-Efficient Deep Learning Training Framework by Using Error-Bounded Lossy Compression

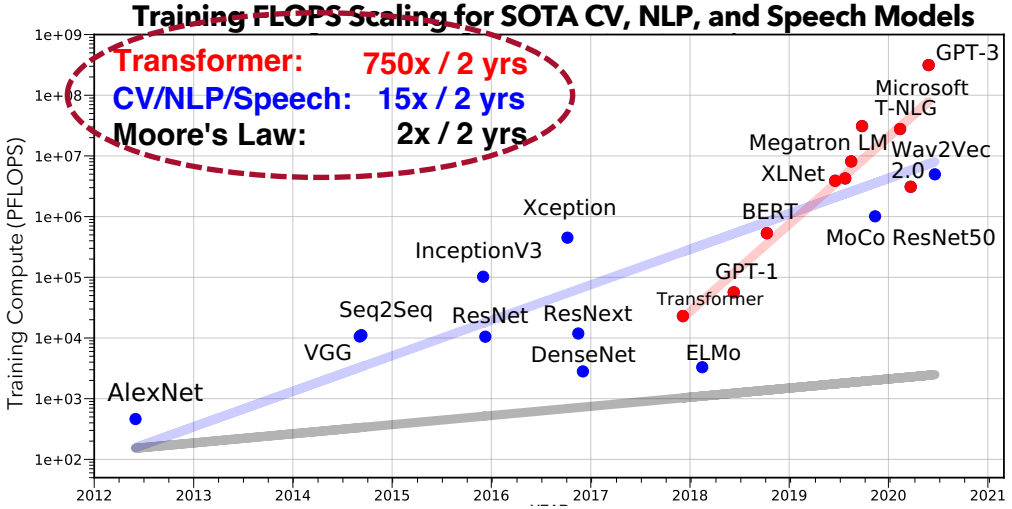
Published in *Proceedings of the VLDB Endowment, Vol. 15, No. 4, 2021*

System Issues with Large AI Models

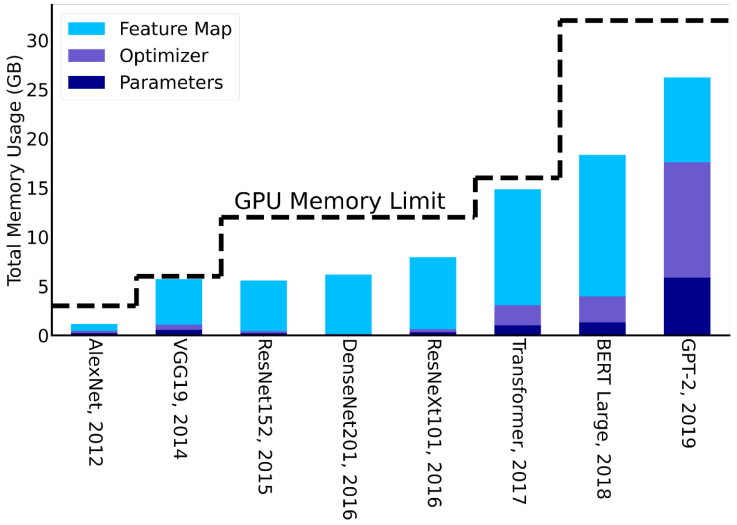
LUDDY
SCHOOL OF INFORMATICS, COMPUTING, AND ENGINEERING

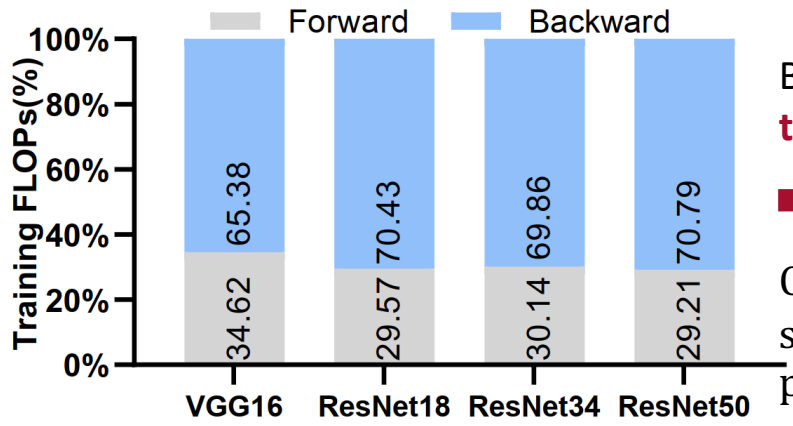
Trend





Issue





Solution

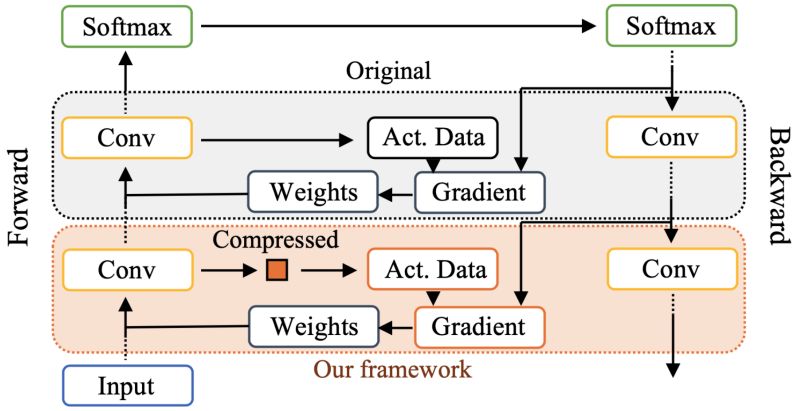
Highly **limited** GPU memory space but **larger** batches are needed [You et al., SC'19]

Our solution [Jin *et al.*, VLDB'22] reduces memory consumption by up to **13.5×**.

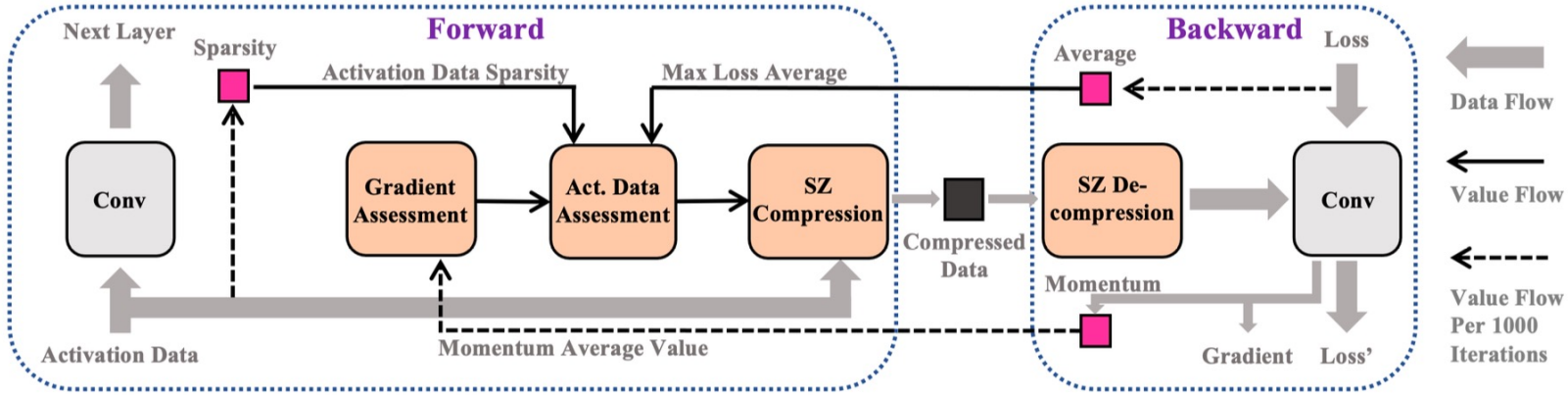
Backward phase consumes **more than 70%** of overall training FLOPs

Our solution [Zhang *et al.*, ICS'21] saves end-to-end training-and-pruning time by **up to 2.3×**.

System Design



Data flow in a sample iteration of training CNNs



Overview of our proposed memory-efficient DNN training framework - COMET

➢ **Activation Data Storage in Training**

- Must being stored until used in back propagation
- Long waiting period between generating and using the data

- **Parameter collection**: collect parameters for analysis and updating compression configurations
- **Gradient assessment**: estimate acceptable σ in the gradient
- **Activation assessment**: estimate acceptable error bound for compressing activation data
- **Adaptive compression**: deploy lossy compression

Design Detail

- **Parameter Collection**
  - **Offline parameters**: batch size, activation data size, corresponding output layer size
  - **Simi-online parameters**: activation data sparsity, average loss, average momentum value
- **Gradient Assessment**
  - Compute σ based on parameters and empirical experience:

$$\sigma = 0.01 M_{Average}$$

- **Activation Assessment**
  - Compute error bound based on parameters and theoretical analysis:

$$eb = \frac{\sigma}{a\bar{L}\sqrt{NR}}$$

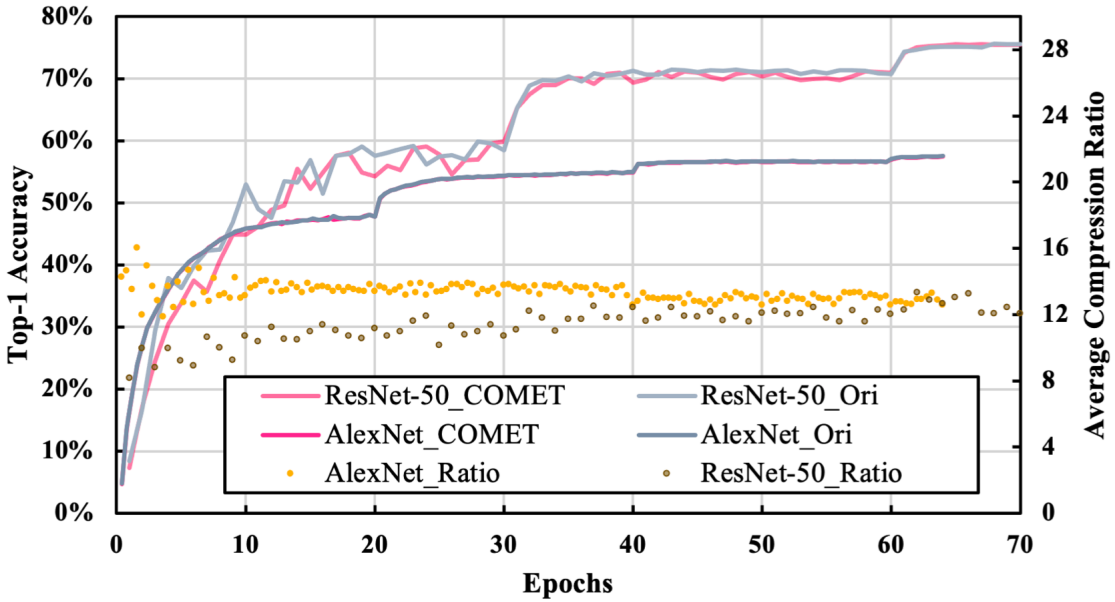
- **Adaptive Compression**
  - Compression configuration update every 1000 iterations
  - Modified **cuSZ** for compressing sparse **floating-point** data

Memory Usage Evaluation

LUDDY
SCHOOL OF INFORMATICS, COMPUTING, AND ENGINEERING

➢ Memory Footprint Reduction

- High compression ratio, **up to 13.5x**
- **Little/no** testing accuracy loss



Training accuracy curve comparison between the baseline and our proposed framework.

Neural Nets		Top-1 Accuracy	Peak Mem.	Max Batch	Conv. Act. Size	COMET	JPEG-ACT
AlexNet	b.	57.41%	2.17 GB	512	407 MB	13.5×	-
	c.	57.42%	0.85 GB	2048	**30 MB**		
VGG-16	b.	68.05%	17.29 GB	64	6.91 GB	11.1 ×	-
	c.	68.02%	5.04 GB	256	**0.62 GB**		
ResNet-18	b.	67.57%	5.16 GB	256	1.71 GB	10.7 ×	7.3 ×
	c.	67.43%	1.37 GB	1024	**0.16 GB**		
ResNet-50	b.	75.55%	15.57 GB	128	5.14 GB	11.0 ×	6.0 ×
	c.	75.51%	4.40 GB	512	**0.46 GB**		

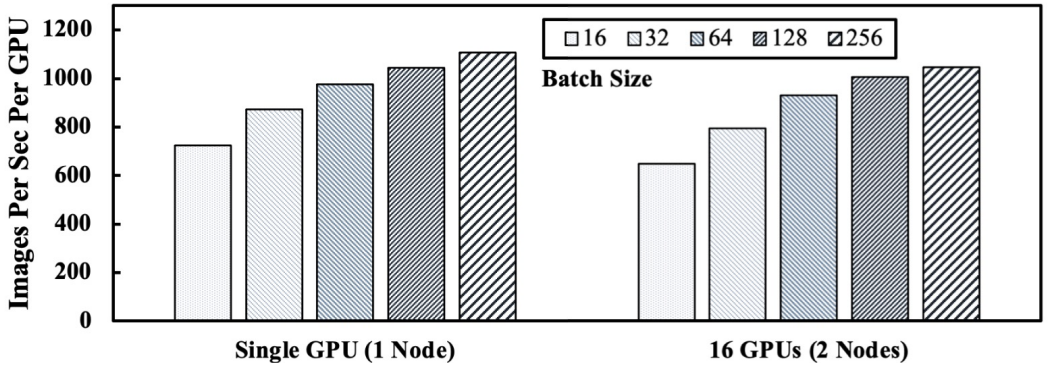
b.= baseline, c.= compressed

Comparison of accuracy and activation size between baseline training and our proposed framework

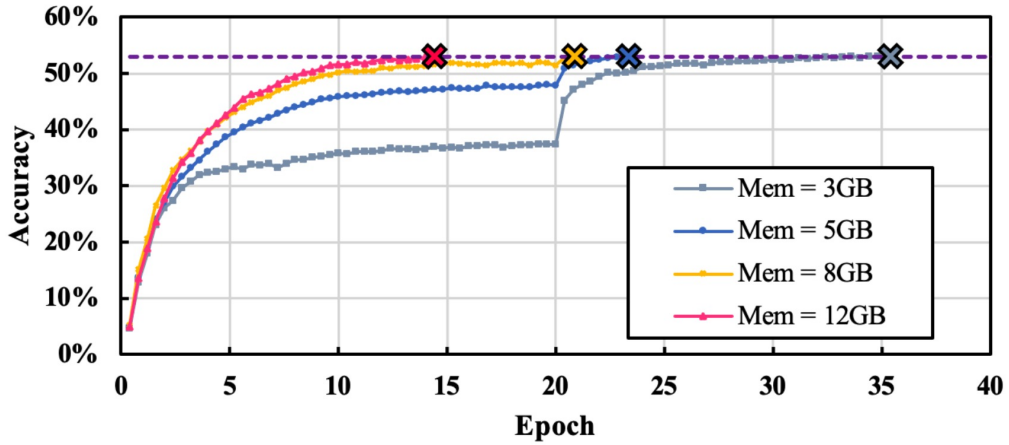
Performance Evaluation

➢ Performance Improvements

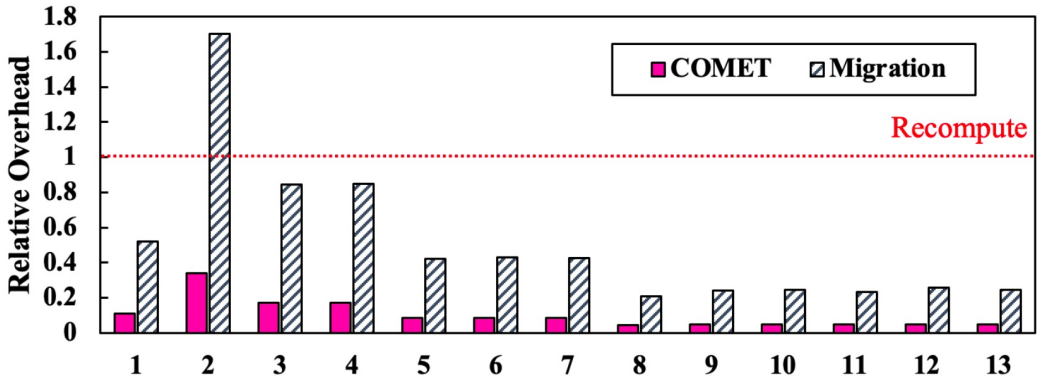
- Low compression overhead, significantly lower than data migration solution (e.g., **7%** on VGG-16)
- High raw throughput (sample/sec) improvement with better resource utilization (e.g., **1.24x** on ResNet-50)
- End-end performance improvement: train model faster (e.g., **2x** on AlexNet)



Training performance on ResNet-50 with different Batch size



Validation accuracy curve of COMET under different GPU memory constraint on AlexNet



Overhead comparison between migration, recomputation

Q&A

Thank You!

Email: ditao@iu.edu
Website: https://www.dingwentao.com/

Acknowledgements