waveSZ: A Hardware-Algorithm Co-Design of Efficient Lossy Compression for Scientific Data

Jiannan Tian	*The University of Alabama*
Sheng Di	*Argonne National Laboratory*
Chengming Zhang	*The University of Alabama*
Xin Liang	*University of California, Riverside*
Sian Jin	*The University of Alabama*
Dazhao Cheng	*University of North Carolina at Charlotte*
Dingwen Tao	*The University of Alabama*
Franck Cappello	*Argonne National Laboratory*

February 24, 2020 PPoPP '20 at San Diego, California, USA

THE UNIVERSITY OF ALABAMA® Argonne NATIONAL LABORATORY

Trend of Supercomputing Systems

Storage capacity and bandwidth are developing **more slowly** compared to computational capability.

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

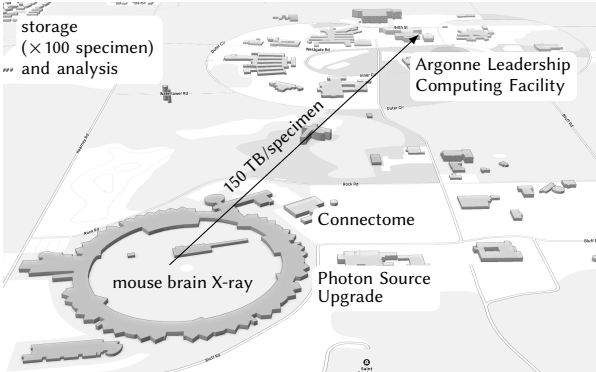
Source: F. Cappello (ANL)

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

Current Status of Scientific Applications

Today's scientific research is data driven at a large scale (simulations or instruments). PB to process & analyze. (PB datasets are coming.) Data reduction is on demand.

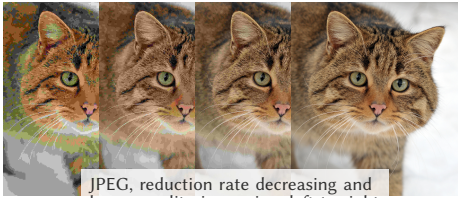
- **cosmology simulation** HACC [a] generates ❶ **20 PB** data per one-trillion-particle (10^{12}) simulation, ❷ exhausting the FS [b] and ❸ taking long to store [c]. ❹ Reduction at rate **10** needed.
- **climate simulation** CESM generates ❶ **1 TB** data per compute day, ❷ increasing hardware budget in storage (NCAR), from 20% (2013) to 50% (2017).❸ Reduction rate at **10+** needed [A. Baker et al., HPDC '16].
- **APS-U Project** (High-Energy X-ray Beams Experiments) brain initiatives: ❶ **multi-hundred PB** of storage. ❷ Data analysis performed off-site on ANL Mira, with connection at 100 GB/s [d,e]. ❸ Reduction rate at **100** needed.



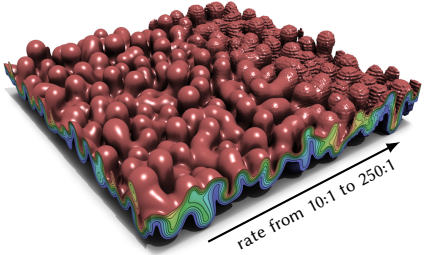
a Hardware/Hybrid Accelerated Cosmology Code
b Mira at ANL has 26 PB FS, 20 PB/26 PB ≈ 80%
c NSF Blue Waters (1TB/s I/O bandwidth), 5h30m to store the data
d Would take ∼115 days to transfer the data
e There is no 100 PB buffer at the APSL :(

(Error-Bounded) Lossy Compression Matters

- Scientific datasets lossless-compressed at rate **2:1** [Son et al. 2014]
  - represented in floating-point
  - We need **10:1** or even higher!
- Industry lossy compressors offer much higher reduction rate.
  - designed/optimized considering human perception
  - not suitable for supercomputer applications
- **Strict** error control toward scientific **discovery** 🚀 and accurate **postanalysis** 💡
  - data analysis with lossy datasets (after or during simulation)
  - execution restarting from failures
  - calculation from lossy data in memory
- Need diverse compression modes
  - absolute error bound (L^∞ norm error)
  - pointwise relative error bound
  - RMSE error bound (L^2 norm error)
  - fixed bitrate
- **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

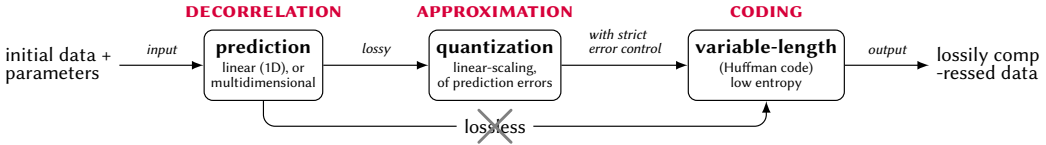


JPEG, reduction rate decreasing and hence quality increasing, left to right

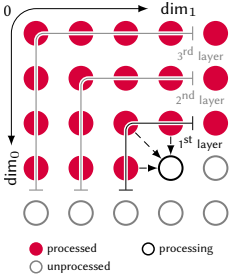


lossy compression for scientific data at varying reduction rate
figure from Peter Lindstrom, LLNL

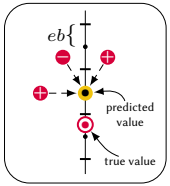
How SZ Works



- Lorenzo predictor allows arbitrary-dimensional prediction.
  - $\sum_{0 \le k_{1\ldots d} \neq \mathbf{0}}^{k_{1\ldots d} \le n} \left\langle \prod_{j=1}^{d} (-1)^{k_j+1} \binom{n}{k_j} \right\rangle \cdot D_{x_1-k_1,\cdots,x_d-k_d}$.
  - Single-layer form Lorenzo predictor works the best generally [Tao et al. 2017]. 2D form:
    $\ell(D_{0,0}) = \mathrm{dot}\left(\begin{bmatrix} -1 & 1 \\ 1 & 0 \end{bmatrix}, \begin{bmatrix} D_{-1,-1} & D_{0,-1} \\ d_{-1,0} & D_{0,0} \end{bmatrix} \right)$.
- Customized Huffman encoding
  - `sizeof(T)`-byte long symbol to Huffman code
  - high quantization quality (aggregated in center) makes Huffman coded bitstream more possible to further `gzip`



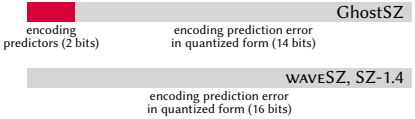
Lorenzo (ℓ) prediction



linear-scaling quantization

Issues with SZ and Its Current FPGA Implementation

- Low throughput of SZ
  - lack of parallelism: SIMD and SIMT cannot apply
- Limitations in FPGA GhostSZ
  - totally performance-driven design
  - 3 predictors in use, need extra bits to encode
  - **more** "workflow pipelines" (more resource)
  - **low** compression ratio



- New use scenarios of adopting FPGA
  - real-time processing; "inline processing" (Intel, 2018)
  - *ExaNet—an FPGA-based direct network architecture of the European exascale systems* [Ammendola et al. 2018]

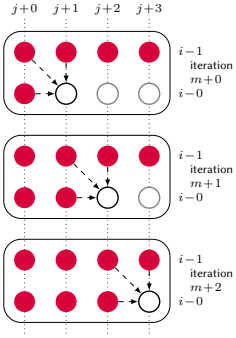


Figure 1: Loop-carried dependencies due to writeback.

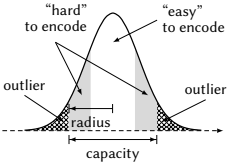


Figure 2: General distribution pattern of quantization code.

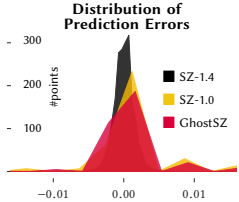
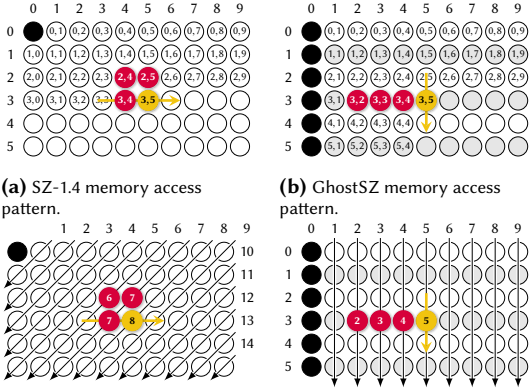


Figure 3: CESM-ATM CLDLOW.

Memory Access Pattern and Dependency



(a) SZ-1.4 memory access pattern.

(b) GhostSZ memory access pattern.

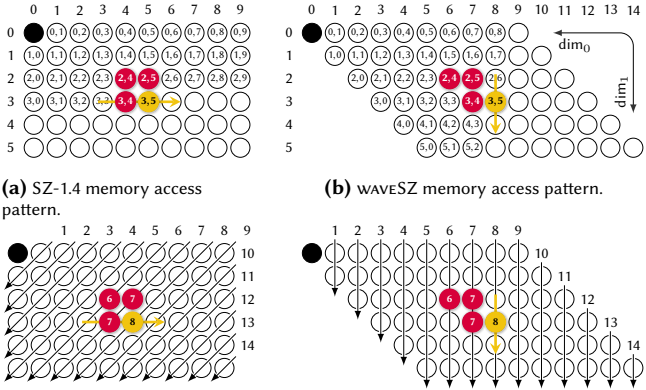
(c) SZ-1.4 dependency in Manhattan distance.

(d) GhostSZ dependency in Manhattan distance.

Figure 4: SZ-1.4 and GhostSZ: memory access pattern and data dependency in Manhattan distance.

- Dependencies denoted with Manhattan distance from ● zero point
- SZ-1.4
  - iterate against the dependencies, see Fig. 4(c)
  - RAW at the last cycle, impossible to extract parallelism
- GhostSZ
  - overlook multidimensional smoothness
  - slice data of dimensionality into 1D
  - hence multiple ● zero points
  - no dependency "vertically"

Memory Access Pattern and Dependency (cont'd)



(a) SZ-1.4 memory access pattern.

(b) waveSZ memory access pattern.

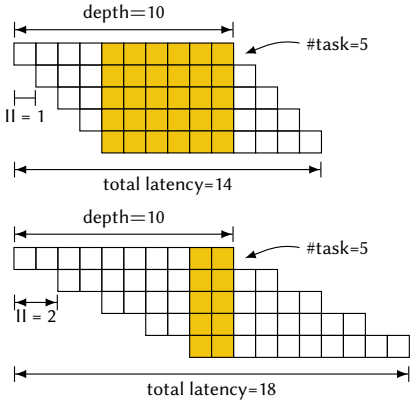
(c) SZ-1.4 dependency in Manhattan distance.

(d) waveSZ dependency in Manhattan distance.

Figure 5: SZ-1.4 and waveSZ: memory access pattern and data dependency in Manhattan distance.

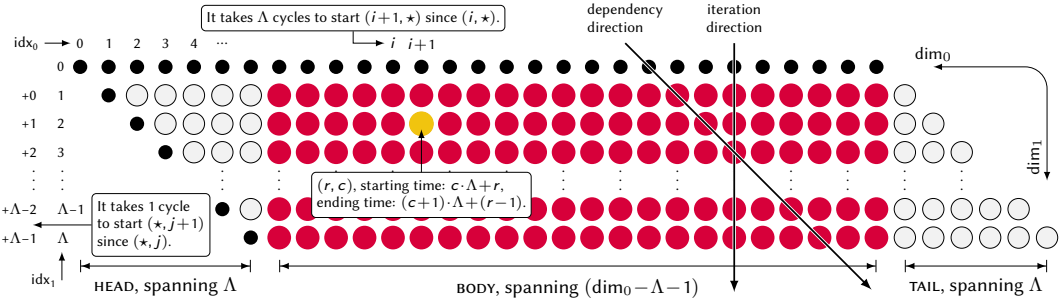
- Dependencies denoted with Manhattan distance from • zero point
- waveSZ
  - iterate along the aligned dependency-free points
  - exploit the parallelism by pipelining
- Pipelining
  - not to change too much
  - expect platform-support pipelining control

Explicit Pipeline



- How pipeline works
  - *depth*: #cycles to complete one iteration
  - *initiation interval* (II): #cycles to wait until the next iteration
  - total latency = $(\#\text{task} - 1) \times \text{II} + \text{depth}$
  - speedup = $(\text{depth} \times \#\text{task})$ / total latency
  - II=1 $\rightarrow$ II=2, speedup: 2.78× $\rightarrow$ 3.57× (28% better) for this demo.
  - Suppose #task $\rightarrow \infty$, speedup = $\frac{\text{depth} \times \#\text{task}}{(\#\text{task}-1) \times \text{II} + \text{depth}} \sim \frac{\text{depth}}{\text{II}}$, hence, II matters.
- Reason why is not (always) II=1
  - data dependency (loop carried)
  - resource in use (e.g., memory port)

Temporal-Spatial Mapping



- FPGA + wavefront memory layout = more pipelining control
- Ideally, suppose Λ cycles to finish (prediction + quantization), no stall if ❶ II = 1, and ❷ (vertically) iterating over Λ points from (r, c) to $(r, c+1)$
- BODY ("perfect loop") unrolled with factor Λ (= vertical dimension) and II = 1

Performance

- Platform
  - target board: Xilinx ZC706
  - programming: C/C++ and high-level synthesis
  - HLS: C/C++ semantics to HDL
  - unroll with II=1, loops become pipelined commands
- Datasets
  - *Scientific Data Reduction Benchmarks* (SDRB) suite from `https://sdrbench.github.io`
  - 3 representative datasets, a diversity of fields
- Synthesis report
  - successfully unroll body, the "perfect loop" with II=1
  - waveSZ use less resource

	# fields	type	dimensions	example fields
CESM-ATM	79	`float32`	1800×3600	`CLDHGH, CLDLOW`
Hurricane	20	`float32`	100×500×500	`CLOUDf48, Uf48`
NYX	6	`float32`	512×512×512	`baryon_density`

Figure 6: Representative datasets.

	total	waveSZ	(%)	GhostSZ	(%)
BRAM_18K	1090	9	0.84	162	14.86
DSP48E	900	0	0.00	63	7.00
FF	437,200	4473	1.02	19470	4.45
LUT	218,600	8208	3.75	27030	12.37

Table 2: Resource utilization from synthesis.

Performance (cont'd)

- Baseline: CPU SZ-1.4, and GhostSZ
- Multilane waveSZ on FPGA vs OpenMP SZ
- Compressor configuration
  - error bound set to 10^{-3} relative to value range
  - 16-bit quantization code, waveSZ, ompSZ
  - 14-bit quantization code, 2-bit predictor code for GhostSZ
- Performance, in MB/s

	waveSZ	GhostSZ	SZ-1.4
CESM-ATM	995	130	114
Hurricane	838	101	122
NYX	986	110	125

- Scaling up
  - OpenMP parallelizes sublinearly, 59% at 32 cores
  - OpenMP version support 3D only
  - FPGA implementations saturate at PCIe bandwidth

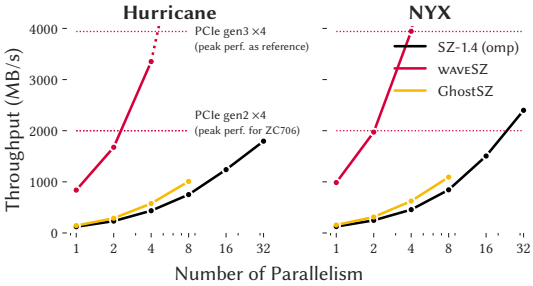


Figure 7: Performance.

Statistics and Postanalysis

Compression Ratio (CR)

- How CR is affected
  - distribution of quantization code
  - the amount of the unpredictable/"outliers"
  - how it is losslessly encoded
- GhostSZ reserves 2 bits to encode predictors in use, diverging 3 peaks at `0b00...`, `0b01...`, `0b11...`.
- Techniques in use
  - G* stands for `gzip` only,
  - H*G* stands for Huffman + `gzip`.
  - With G*, waveSZ shows higher CR than that of GhostSZ.
  - With simulated H*G*, waveSZ CR ≈ SZ-1.4 CR.

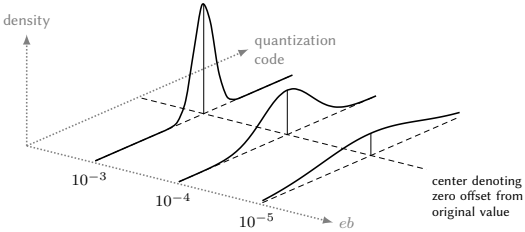
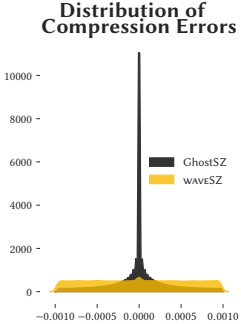


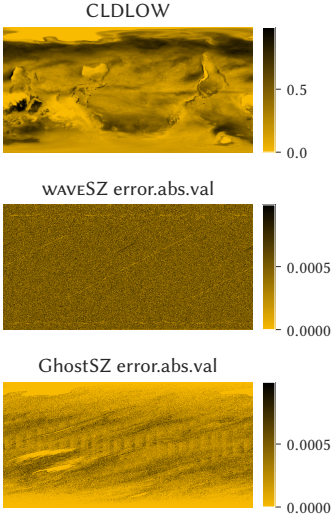
Figure 8: Error bound changing impacts on CR.

		CESM-ATM	Hurricane	NYX
GhostSZ		7.9	6.2	6.6
waveSZ	G*	12.3	13.2	18.3
	H*G*	29.4	20.3	34.8
SZ-1.4		31.2	21.4	33.8

Table 3: Compression ratio.

Statistics and Postanalysis (cont'd)





- $eb = 10^{-3}$ relative to value range
- GhostSZ has slightly higher PSNR
  - Curve fitting is more intuitive.
  - Lorenzo (multidimensionally linear) has lower chance to get high prediction accuracy in the similar-value areas.
  - in the case of CESM-ATM
- Tradeoff between these two predictors
  - multidimensionality (Lorenzo)
  - higher PSNR (curve fitting)
  - less resource use (Lorenzo)
  - higher CR (Lorenzo)

	GhostSZ	waveSZ	SZ-1.4
CESM-ATM	73.9	65.1	64.9
Hurricane	70.6	66.0	65.0
NYX	74.5	66.5	65.2

Table 4: PSNR.

Conclusion and Future Work

Conclusion

- We adopt a wavefront memory layout to alleviate dependency SZ-1.4 with arbitrary-dimensional predictor.
- We propose a co-design framework for SZ lossy compression, waveSZ, and implement it in HLS.
- We propose a hardware-algorithm co-optimization (e.g., via HLS directive, base-two algorithmic operations).
- We evaluate on three real-world datasets from SDRB suite, showing 2.1× CR and 5.8× througuput on average, compared with the current FPGA implementation.

Future Work

- Integrate open-source production-level `gzip`
- Integrate Huffman encoding

Thoughts on Future Systems

- Co-acceleration
  - FPGA is not in place of manycore accelerator
  - manycore + FPGA (the availability)
- What's added
  - feature: low latency (and high throughput)
  - real-time processing in big-data analytics

Acknowledgement

This research was supported by the Exascale Computing Project (17-SC-20-SC), a joint project of the U.S. Department of Energy's Office of Science and National Nuclear Security Administration, responsible for delivering a capable exascale ecosystem, including software, applications, and hardware technology, to support the nation's exascale computing imperative. The material was also supported by and supported by the National Science Foundation under Grant No. 1305624, No. 1513201, and No. 1619253.

Thank You !

ANY QUESTION? 🧐

THE UNIVERSITY OF ALABAMA

Argonne NATIONAL LABORATORY

ECP EXASCALE COMPUTING PROJECT

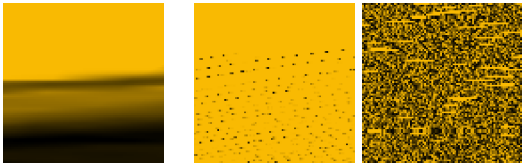
U.S. DEPARTMENT OF ENERGY

BackUp (Rasterization)

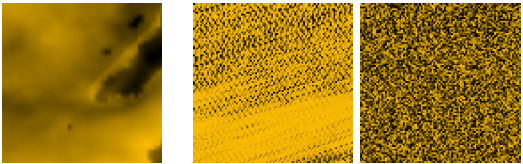
Due to the rasterization (an 1800×3600 datum is visualized with serval inches each dimension), the compression error for Lorenzo predictor seems significantly worse than that of Order-{0,1,2}. (top: origin, bottom left: GhostSZ error, bottom right: waveSZ error)

@(900, 0) origin, Ghost.err, wave.err

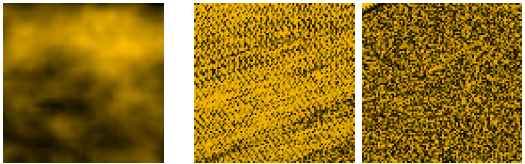
@(1800, 0) 90×90



@(900, 900)

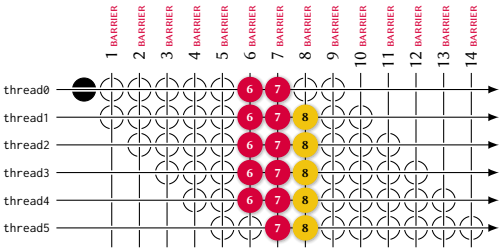


@(1800, 900)



BackUp (FPGA and GPU)

- Prediction + quantization
  - tight dependencies in the original SZ
  - alleviated with wavefront, dependency on one direction left
  - expensive synchronizations across iterations
- Lossless stage
  - we have open-source `gzip`
  - too many if-branches and random accesses



BackUp (PSNR and CR)

waveSZ is using the same predictor as SZ-1.4 mainly but has higher PSNR because

- waveSZ doesn't do any bit truncation to unpredictable data.
- Interestingly, on the NYX dataset, waveSZ has a slightly higher CR ($H^{\star}G^{\star}$). waveSZ goes along "y-direction" (corresponding to outer loop) to overlap the prediction and quantization latency and then change to the following point in the "x-direction" (as shown in Slide 10).

BackUp (ℓ-Predictor)

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

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

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