Lossy Scientific Data Compression With *SPERR*

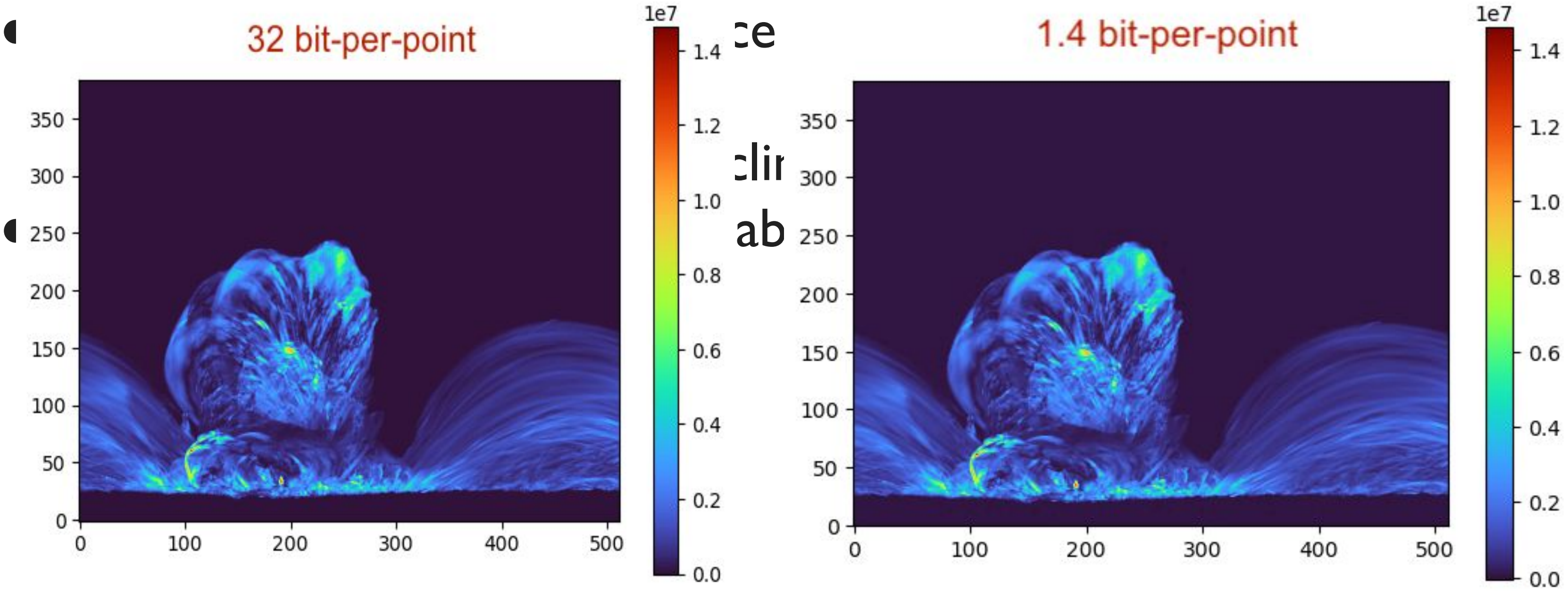
Samuel Li, Peter Lindstrom, John Clyne

(NCAR, LLNL, NCAR)

The 37th IEEE International Parallel &
Distributed Processing Symposium (IPDPS)
May 18 2023, St. Petersburg, FL

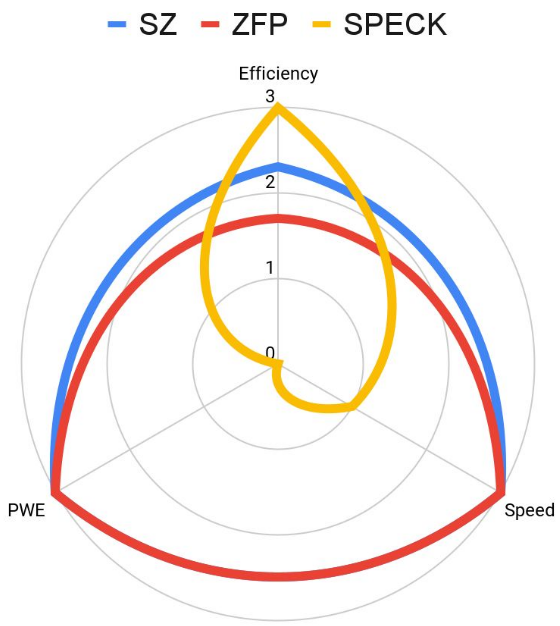
Motivation

- Numerical simulation is critical in many disciplines of scientific research, e.g., climate, combustion, aerodynamics.



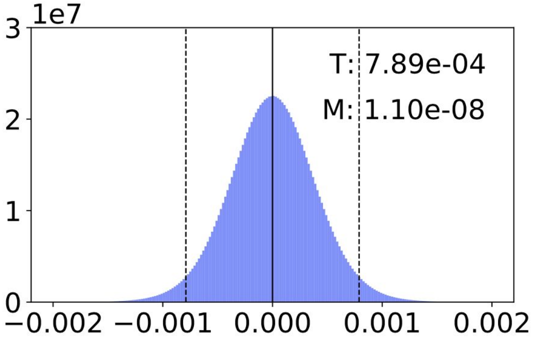
Existing Solutions

- *Lossless* compression
  - < 2X reduction on complex data
- *Lossy* compression
  - ZFP [Lindstrom], SZ [Di et al.], etc.
  - Wavelet (SPECK [Islam et al.], etc.)
- Aspects of lossy compression:
  - Efficiency
  - Speed
  - <u>PWE (pointwise error guarantee)</u>

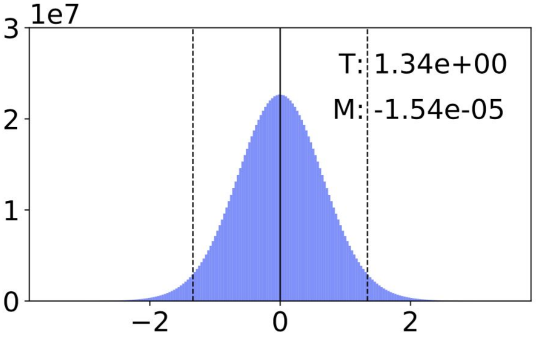


Approach

- <u>SPERR: SPEck with ERRor bounding</u>
- Improve the highly-efficient SPECK algorithm with
  - 1) ability to provide a PWE guarantee
  - 2) faster speed
- Explicit outlier coding:
  - Find all data points exceeding a PWE tolerance (aka. *outliers*).
  - Encode a *corrector* : {pos, val}.
  - Bring outliers into the PWE tolerance with encoded correctors.



(a) X-Vel. before outlier coding

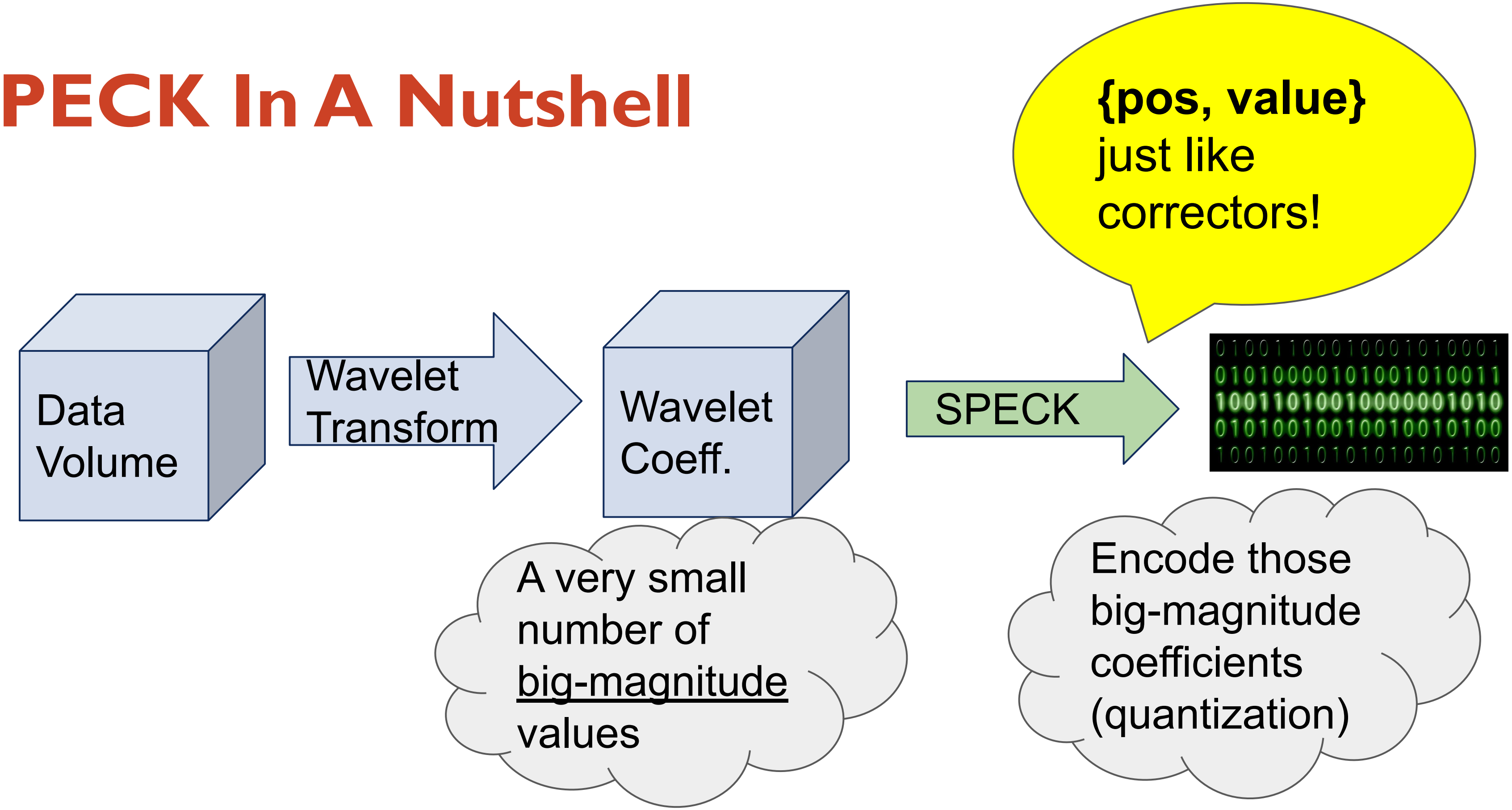


(c) Ens. before outlier coding

Challenges

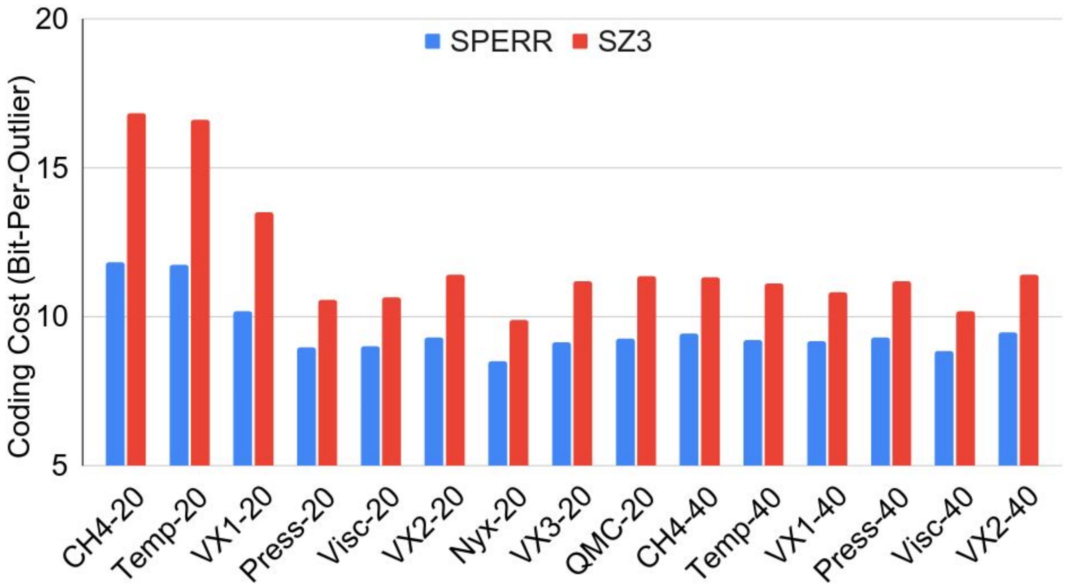
- Challenge 1: perform outlier coding economically.
- Challenge 2: find the right balance between SPECK and Outlier Coding
    - Total storage = SPECK + Outlier Coding
    - Too much SPECK: reduce average error unnecessarily low
    - Too much Outlier Coding: miss out super-high efficiency of SPECK

SPECK In A Nutshell



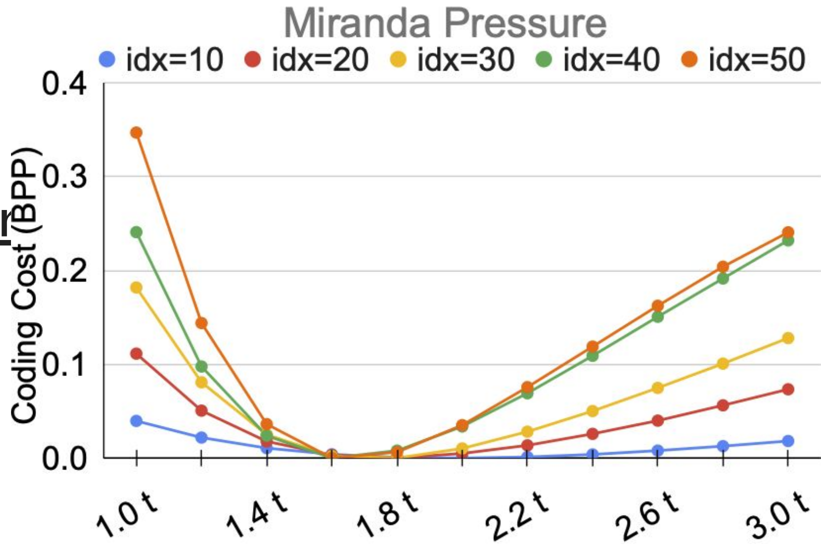
Challenge 1

- Solution: re-use SPECK to encode the correctors.
- A corrector tuple {pos, val} takes <u>64 bits</u> in native storage.
  - ~10 bits with SPECK.
- SZ: the only compressor that also explicitly corrects outliers:



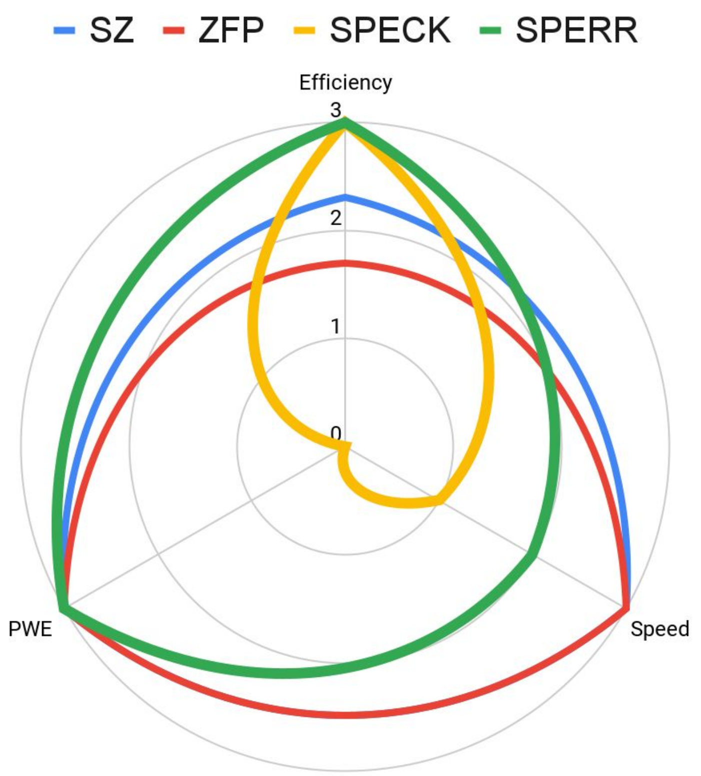
Challenge 2

- Total storage = SPECK + Outlier Coding
  - SPECK: reducing overall error
  - Outlier Coding: bring individual outlier to PWE
- Solution: pre-determine the quantization step size based on the PWE tolerance:
  - q = step size, t = PWE tolerance
  - Smaller q: use more storage on SPECK
  - Bigger q: use more storage on OC



Chosen value: q = 1.5t

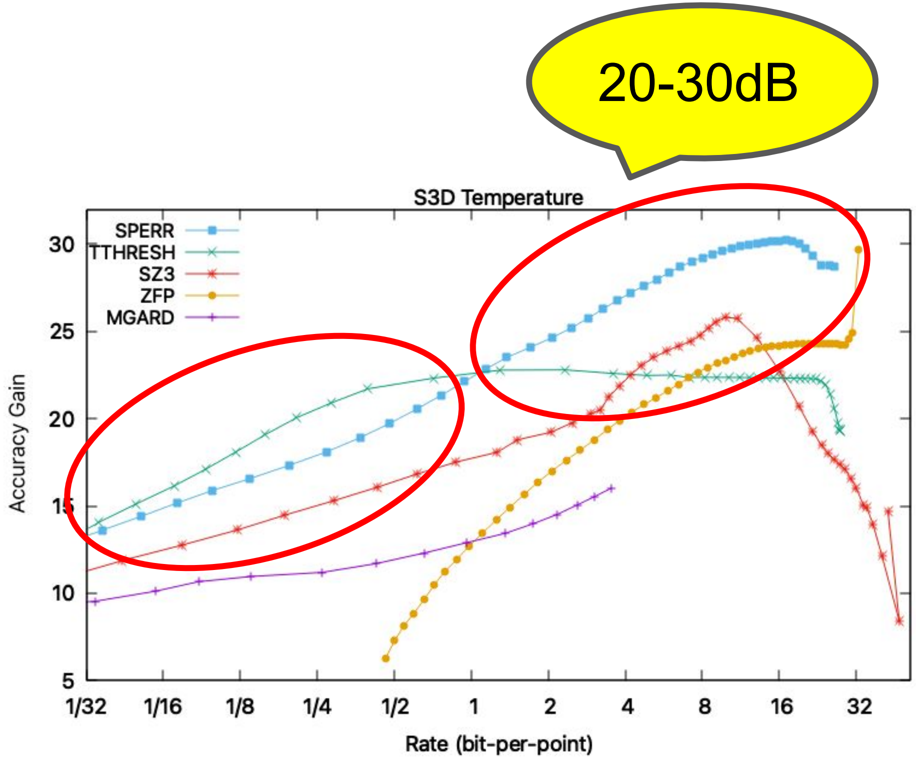
Put Everything Together: SPERR

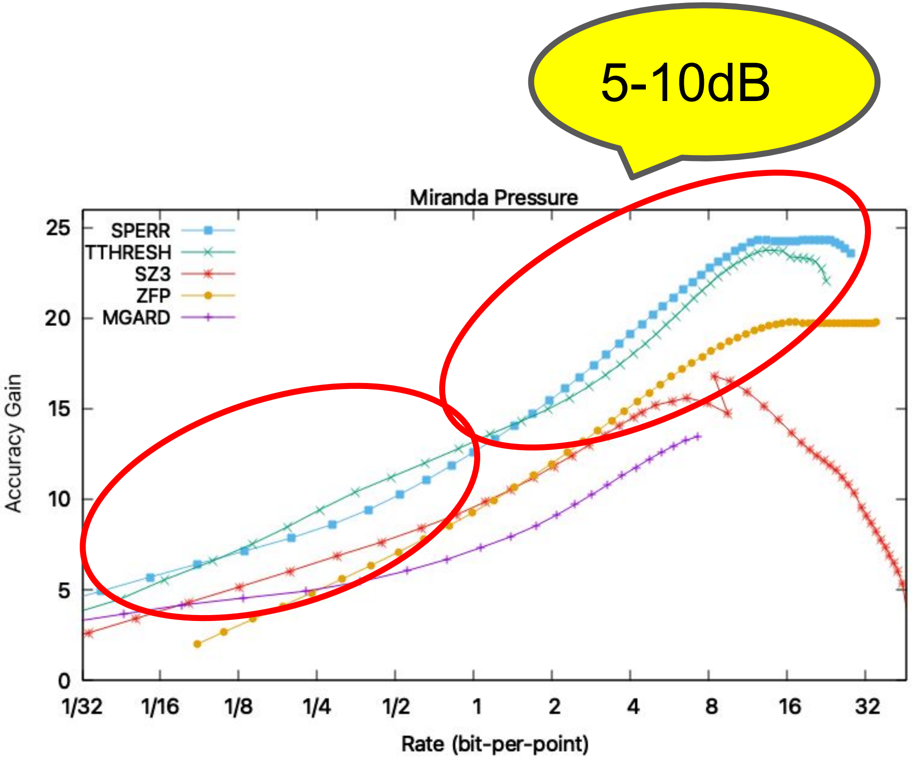


Efficiency Eval. : Rate-distortion Curves

- Plots the efficiency of a compressor at every bitrate.
- Rate: bit-per-point
- Distortion: *__accuracy gain__:* gain = log2(σ / E) - R
    - based on the average error at a certain bitrate
    - measures the amount of info. inferred by the compressor that need not be stored.
    - $gain = \frac{SNR}{20\log_{10} 2} - R \approx \frac{SNR}{6.02} - R$

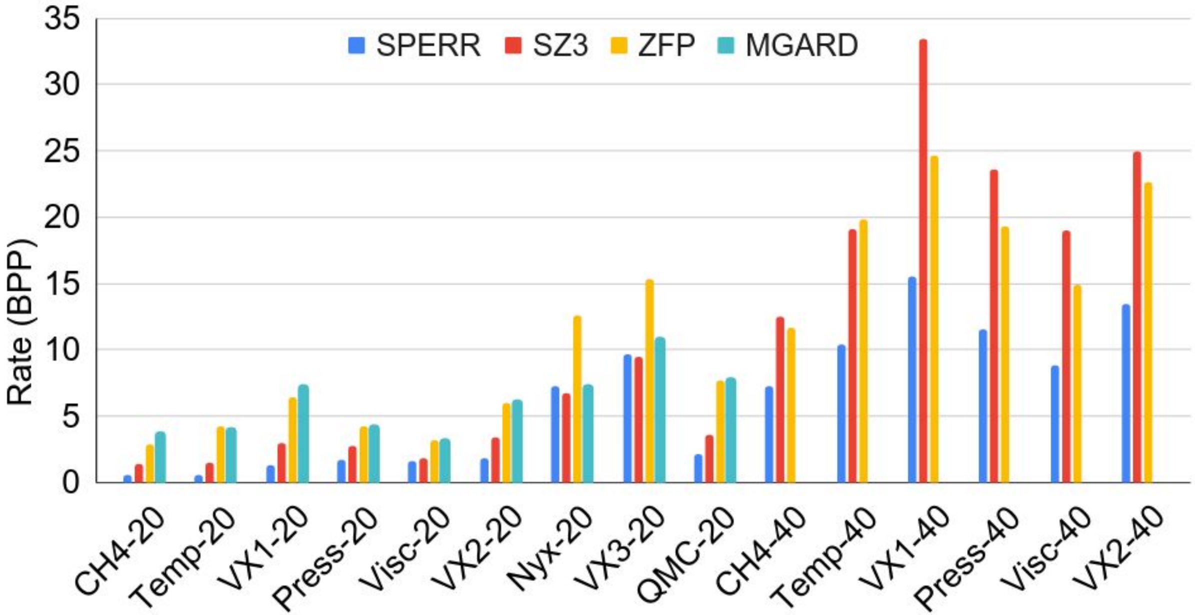
Efficiency: Rate-distortion Curves





Efficiency Eval. : Storage Cost at An Error Tol.

- Given a PWE tolerance, a compressor satisfies it using the least amount storage, *regardless* of the average error.



Conclusion

- SPERR: wavelet-based, yet supports PWE-bounded compression
- SPERR achieves the best compression efficiency among available scientific data compressors.
- It runs slower than SZ and ZFP, which is also the most important area of improvement.