Kai Zhao, Sheng Di, **Robert Underwood**, Hanqi Guo, Fengguang Song, Franck Cappello, and Many More

Argonne NATIONAL LABORATORY

SZ and the FZ Framework

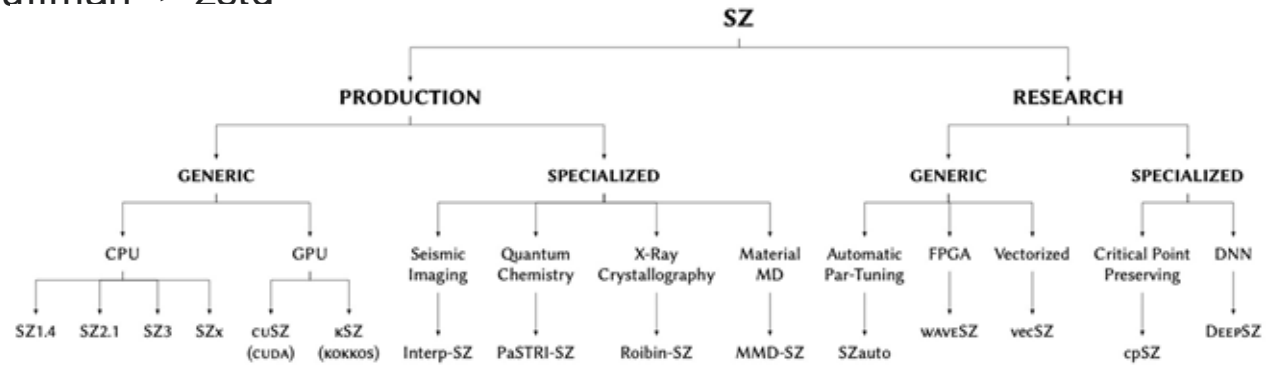
Robert Underwood
Assistant Computer Scientist
Argonne National Laboratory

Scientist at Large, CASE
University of Chicago

U.S. DEPARTMENT OF ENERGY Argonne National Laboratory is a U.S. Department of Energy laboratory managed by UChicago Argonne, LLC.

SZ is a Family of Compressors

- SZ is a family of compressors
  - SZ3, QOZ, MDZ, cuSZ, cuSZp, etc.
- SZ3 is the default (recommended) one for general users on CPU
  - SZ3 has high compression ratios and speed on CPU
  - SZ3 has a modularized design, you can customize it for your own use cases
  - The default compression workflow is like: interpolation -> quantization -> huffman -> zstd



FZ is the Next Generation of SZ: Project Overview

1. Compiler
   a. Enables composition of modules
   b. Enables generation of new modules on the fly
2. Modules Library
   a. GPU
   b. CPU
3. Interface
   a. Visual Analytics Tools
   b. Ease of Use
   c. (Upcoming) Optimization

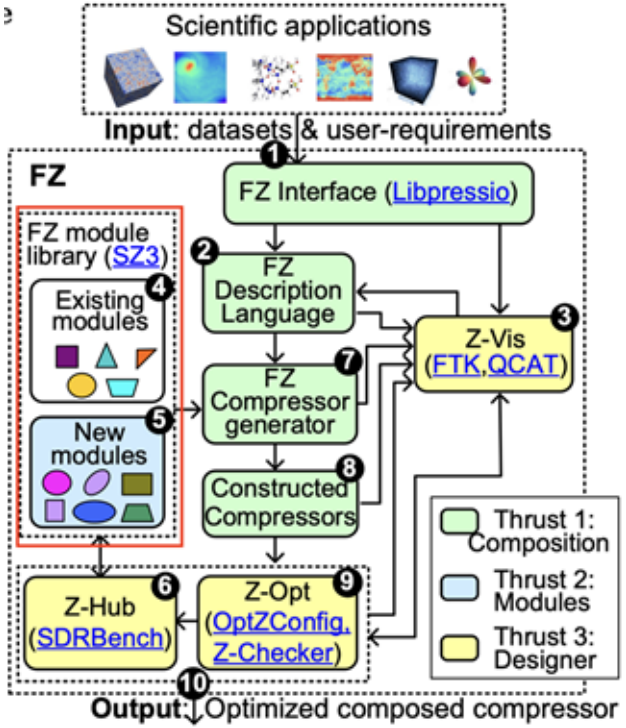


Figure 5: FZ design overview.

FZ Demo

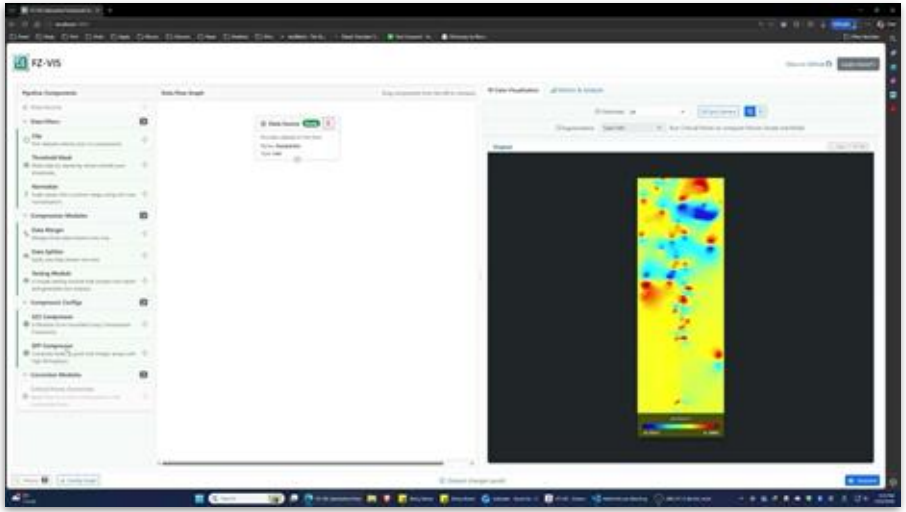
New features

- Critical points and Morse-Smale (MS) segmentation visualization
- MSz integration on critical points and MS segmentation preservation
- Power spectrum visualization

Advantages

- Enable Quantities of Interest (QoI) preservation for the compression
- Help decision making through visualization tools

Demonstrate how the user can achieve the QoI preservation goal (topological features) with the new features



Appendix SZ

Installing SZ

There are four recommended ways to install SZ

- **Spack** develop and use our cutting edge versions
  - `spack repo add https://github.com/robertu94/spack_packages`
  - `spack install sz3# or cusz, cuszp, qoz, etc…`
- **Pip** stable releases of python packages maintained by SZ3 team
  - `pip install pysz`
- **numcodecs-rs** for WASM integration maintained by Juniper Tyree
  - `pip install numcodecs-wasm-sz3`

You can also build from source using cmake. See the README for details

How to compress data with non-finite values

SZ employs an out-of-bounds fallback strategy automatically.

- When a value (like NaN) cannot be meaningfully quantized within the error bound, SZ automatically stores the original FP32 binary representation.

- For datasets with many non-finite values, users may see better ratios by using LibPressio's mask/pre-processing tools to handle these regions before the SZ compression.

How to compress with ABS

MacBookPro:~ kzhao$ ./sz3 -f -i ~/data/hurricane-100x500x500/Uf48.bin.dat -3 500 500 100 **-M ABS 1e-5** -a -o /var/tmp/Uf48.decompressed

compression ratio = 2.26
compression time = 0.511313
compressed data file = /data/hurricane-100x500x500/Uf48.bin.dat.sz.tmp
Min=-53.0225677490234375, Max=39.558197021484375, range=92.5807647705078125
Max absolute error = 1E-05
Max relative error = 1.1E-07
Max pw relative error = 1.8
PSNR = 144.499546, NRMSE= 5.956932725E-08
normError = 0.027575, normErr_norm = 0.000001
acEff=1.000000
compression ratio = 2.261395
decompression time = 0.854659 seconds.
decompressed file = /var/tmp/Uf48.decompressed

How to use PW_REL bounds

SZ3 doesn’t support PW_REL natively, please use LibPressio or SZ2 for such cases.

Limitations

- SZ3 is mainly designed for structured mesh in 1D to 4D, and in FP32 or FP64 format.
- For unstructured mesh, you may compress it as a 1D array.
- For particles/point cloud, we have special compressor that has much better compression ratios (https://github.com/hpdslab/LCP)
- For higher dimension input, you may flat it to 4D for SZ3, but it may not have the optimal compression ratio