Approachable Error Bounded Lossy Compression

Robert Underwood, Ph.D. Candidate, School of Computing

SC20 Everywhere we are | more than hpc.

#ClemsonSC20

CLEMSON UNIVERSITY

School of COMPUTING Clemson University

About Me — Robert Underwood

- Ph.D. Candidate
- Computer Science
- Research Interests:
  - Lossy Compression
  - Reliability and Fault Tolerance
  - High Performance Computing

Personal Website

My CV

Exascale HPC Needs to Process Big Data

- Exascale Apps:
  - CESM-LE – 300TB/instance
  - HACC – 2000PB storage
- Exascale Instruments
  - LCLS-II – >250GB/s steaming

System	Disk Storage	Peak Disk Bandwidth	Memory
Bebop	< 2PB	n/a	0.3 PB
Mira	24 PB	n/a	~ 1 PB
Theta	11PB	n/a	~ 1 PB
Summit	250PB	2.5 TB/s	10PB
Aroura	230PB	25 TB/s	10PB
Projected Exascale	500 PB		

Franck Cappello et al. "Use Cases of Lossy Compression for Floating Point Data in Scientific Datasets". 2018
Machine Characteristics from respective websites accessed 17 September 2020

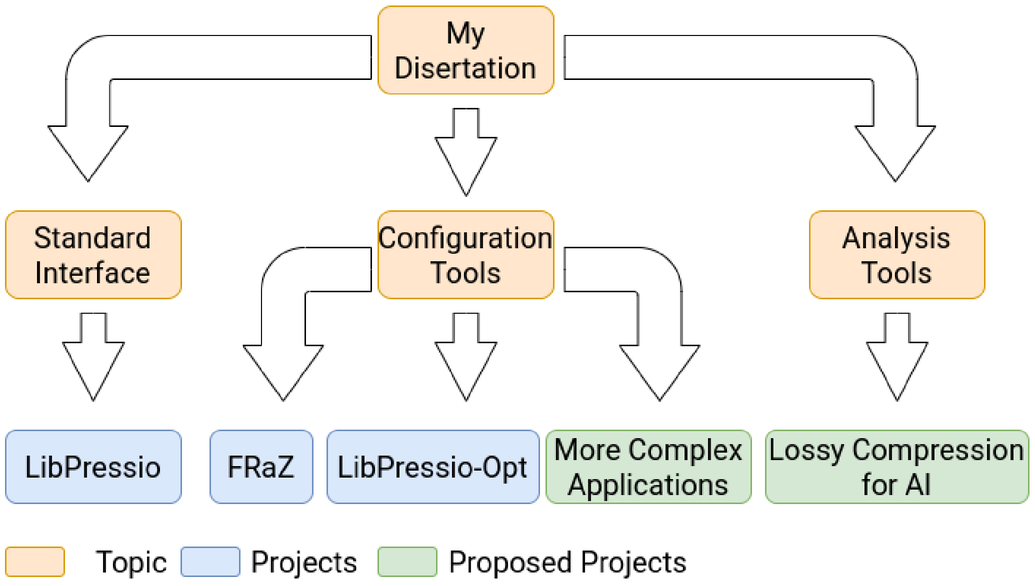
Compression is the Solution

Compression represents data in a more compact fashion

	Lossless	Lossy	Error Bounded Lossy Compression
Examples	ZIP	JPEG	SZ/ZFP/MGARD
Compression Ratio	☹	☺	☺
Ease of Use	☺	☹	??
Data Integrity	☺	☹	☺

Lossy Compression Is Not Approachable

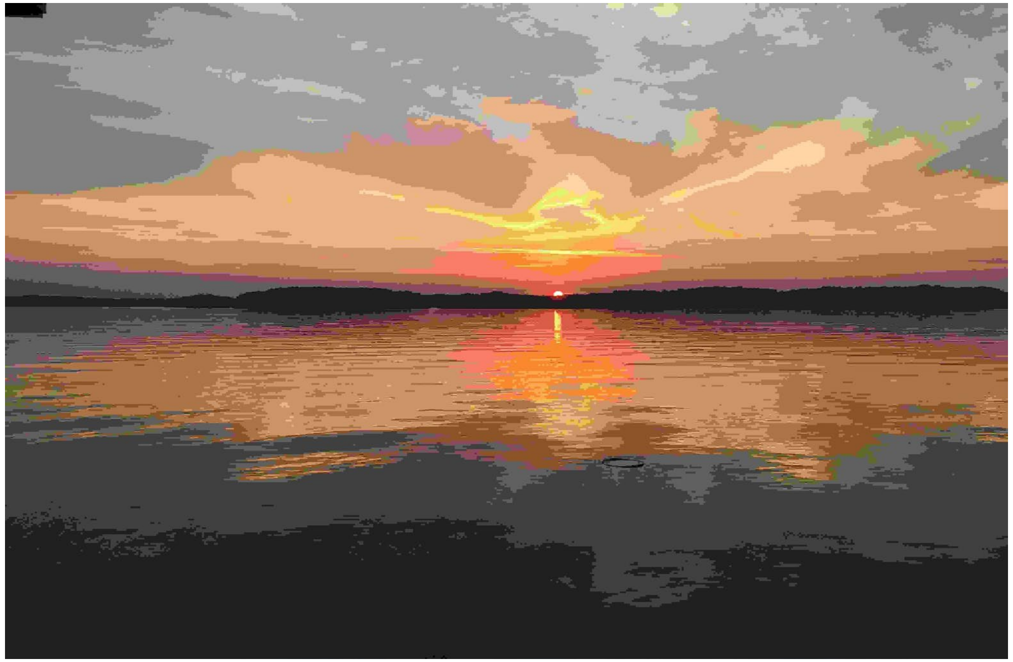
- Too many interfaces
- Too difficult to configure
- Few tools to understand
- **My dissertation provides a single interface to use, configure, understand compression**



Outline

1. Introduction to Compression Principles
2. LibPressio
3. Automated Configuration of Compressors
4. Understanding the Effects of Compression
5. Conclusions and Future Work

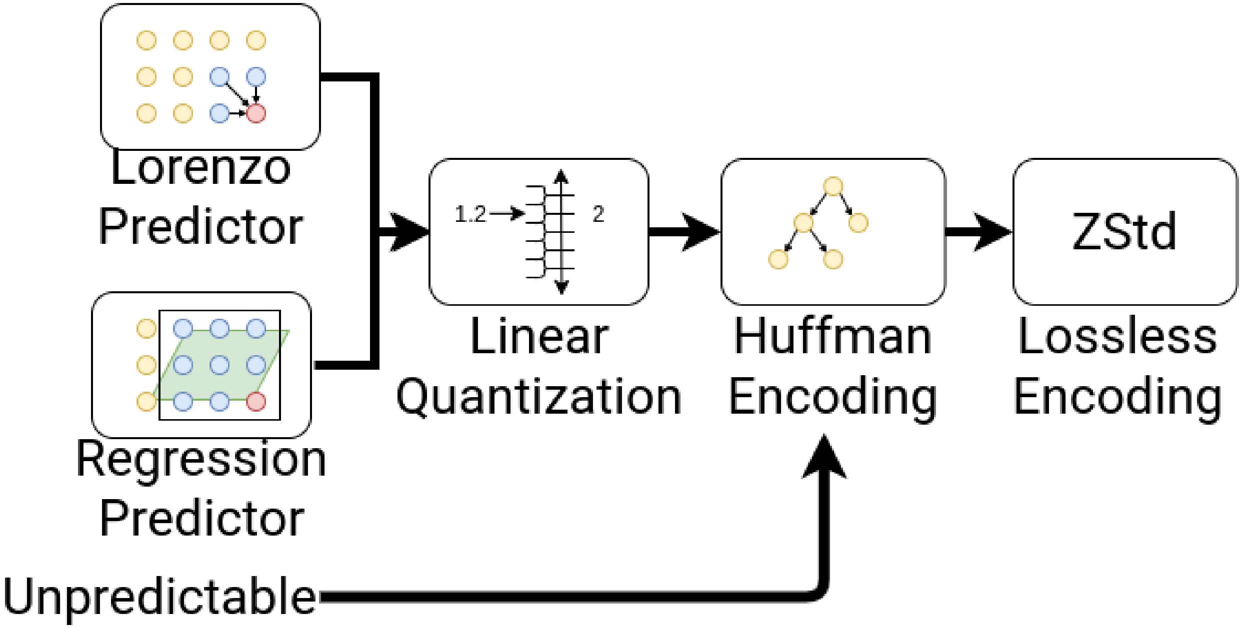
Introduction to Error-Bounded Compression Principles



Lake Hartwell – Lossless (left), Lossy (right). The image on the right is 17 times smaller

SZ

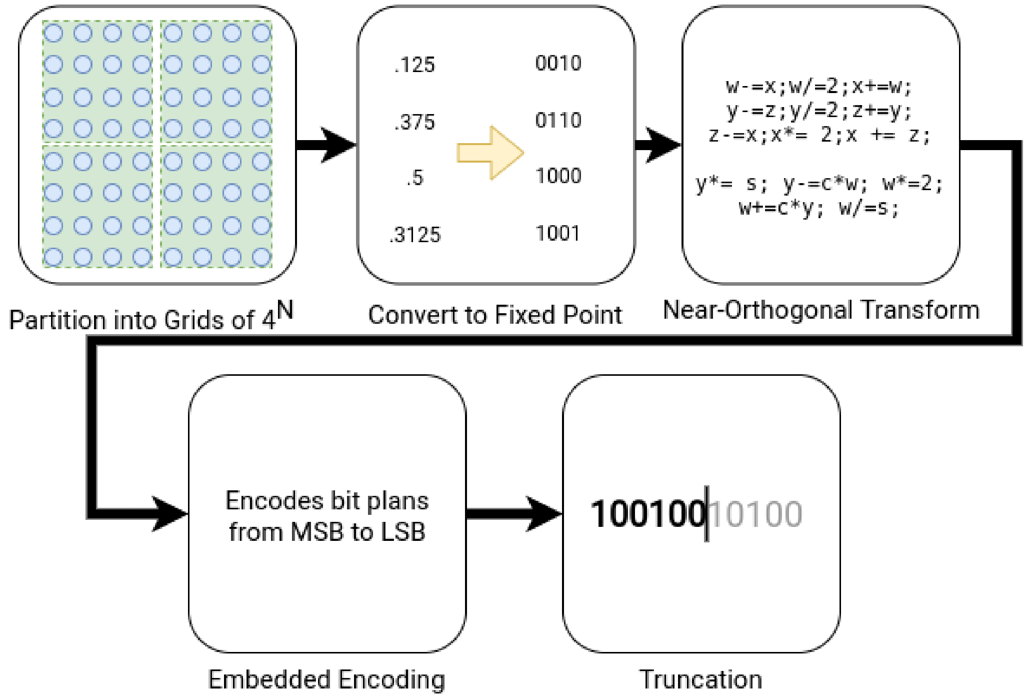
- Prediction Based Compressor
  1. Data Prediction
  2. Linear Quantization
  3. Entropy Encoding
  4. Lossless Encoding



Di, Sheng and Cappello, Franck “Fast Error Bounded Lossy Compression with SZ” 2016

ZFP

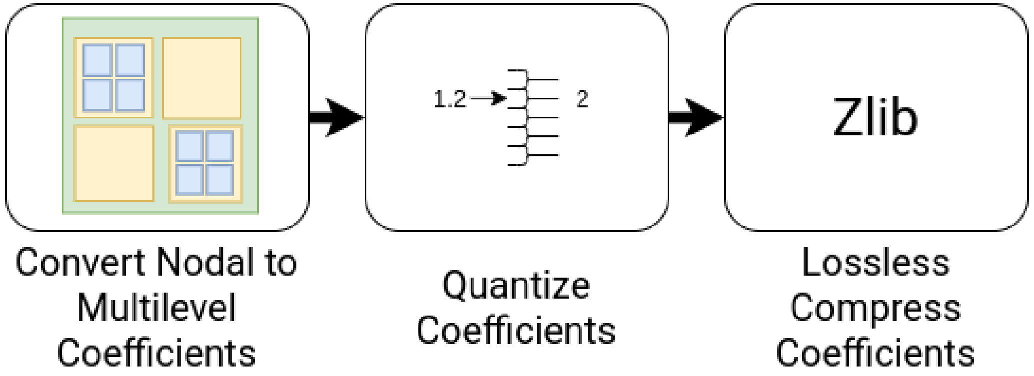
- Transform Based Compressor
  1. Partition into grids of 4^n
  2. Convert to fixed point by block
  3. Near Orthogonal Transform
     - Similar to JPEG Compression
  4. Bit manipulation



Lindstrom, Peter. "Fixed Rate Compressed Floating Point Arrays" 2012

MGARD

- Multi-Grid Method
  1. Determine Multi-grid coefficients
  2. Quantize “binding” coefficients
  3. Losslessly encode quantized coefficients



Whitney, Ben E. “Multilevel Techniques for Compression and Reduction of Scientific Data” 2018

LibPressio

A Generic Abstraction for the Compression of Dense Tensors

LibPressio Provides a Common Interface

- Common Abstractions for:
  - Loading compressors
  - Configuration
  - Compression/Decompression
  - Representing Data
  - Error Reporting
  - Computing Metrics

Get LibPressio

```c
//get the compressor
struct pressio* library = pressio_instance();
struct pressio_compressor* sz = pressio_get_compressor(library,
↪ "sz");

//configure, validate, and assign the options
struct pressio_options* config =
↪ pressio_compressor_get_options(sz);
pressio_options_set_integer(config, "sz:error_bound_mode",
↪ REL);
pressio_options_set_double(config, "sz:rel_err_bound", 0.01);
pressio_compressor_set_options(sz, config);

//read in an input buffer
size_t dims[] = {500,500,100};
struct pressio_data* description =
↪ pressio_data_new_empty(pressio_float_dtype, 3, dims);
struct pressio_data* input_data =
↪ pressio_io_data_path_read(description, "CLOUDf48.bin.f32");

//create output buffers
struct pressio_data* compressed_data =
↪ pressio_data_new_empty(pressio_byte_dtype, 0, NULL);
struct pressio_data* decompressed_data =
↪ pressio_data_new_owning(pressio_float_dtype, 3, dims);

//compress and decompress the data
pressio_compressor_compress(sz, input_data, compressed_data);
pressio_compressor_decompress(sz, compressed_data,
↪ decompressed_data);
```

Current Plugins and Tools

Compressors	Meta-Compressors	Metrics	IO	Tools
• blosc • imagemagick • **fpzip** • **mgard** • **sz** • **zfp**	• linear_quantizer • many_dependent • many_independent • resize • transpose • opt • no-op • **fault_injector** • **random_errors**	• composite (lua) • error_stat • time • size • no-op • **kl_divergence** • **ks_test** • **pearson** • **spatial_error** • **ftk_critical_points** • **external**	• posix • csv • hdf5 • select • empty • no-op	• LibPressio-Tools • LibPressio-Predict • LibPressio.jl • **LibPressio.py** • **Z-Checker**

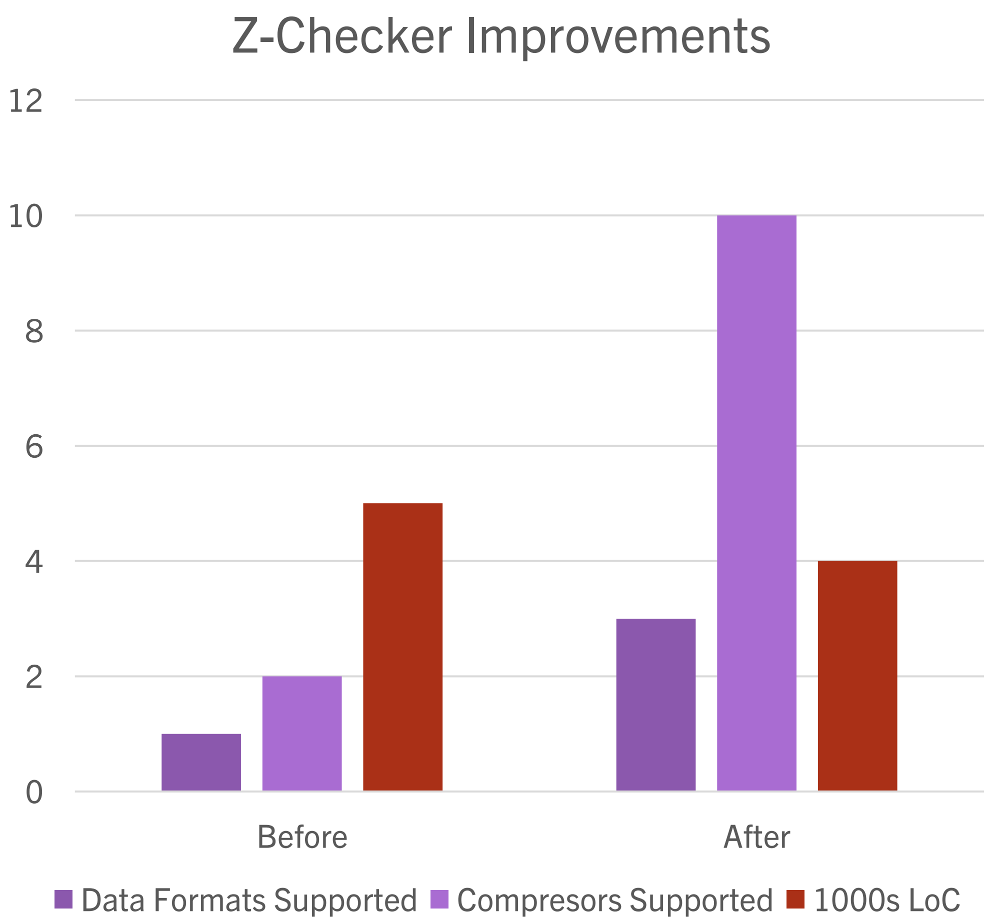
Bolded plugins developed in collaboration with others

Meta Compressors Boost Productivity

- Not a compressor themselves
- Provide common services:
  - Auto Configuration tools
  - Pre/Post Processors
  - Parallel Runtimes

LibPressio Solves Problems

- Writing for multiple compressors is hard: over 100 bugs fixed to date
- Case Study: Z-Checker
  - Save over 1000 LoC (≈21%)
  - Better:
    - Over 10 new compressors
    - Over 3 new data formats
  - Faster: with MPI parallelism
  - Future proof: New compressors just need a recompile



Future Work on LibPressio

- Support for accelerator sharing
  - GPUs
  - Threads
  - FPGAs
- Support for asynchrony/streams
- Support for sparse problems

Automated Configuration of Compressors

Automated Configuration Timeline

IPDPS 2020

FRaZ

- Bounding Compression Ratio

LibPressio-Opt

- Bounding User Metrics
- Performance Improvements

More Complex Apps

- Bounding Complex, Multi-faceted Metrics

1 - FRaZ: Fixed Ratio Compression

Can we tune compression using a control loop to bound the compression ratio?

FRaZ: Approach

Formulate compressor configuration as an optimization problem

$$\max_{\vec{c}\in U} Q\left(d_{f,t}, \widetilde{d_{f,t}}\left(\vec{c};\ \overrightarrow{\theta_c}\right)\right)$$

FRaZ: Approach

Formulate compressor configuration as an optimization problem

Compression Ratio

$$\max_{\vec{c} \in U} Q\left(d_{f,t}, \widetilde{d_{f,t}}\left(\vec{c};\ \overrightarrow{\theta_c}\right)\right)$$

FRaZ: Approach

Formulate compressor configuration as an optimization problem

Compression Ratio

$$\max_{\vec{c} \in U} Q\left(d_{f,t}, \widetilde{d_{f,t}}\left(\vec{c};\ \overrightarrow{\theta_c}\right)\right)$$

Error Bound

FRaZ: Approach

Formulate compressor configuration as an optimization problem

Compression Ratio

$$\max_{\vec{c}\in U} Q\left(d_{f,t}, \widetilde{d_{f,t}}\left(\vec{c};\ \overrightarrow{\theta_c}\right)\right)$$

Error Bound

Allowed Error Bounds

FRaZ: Approach

Formulate compressor configuration as an optimization problem

Compression Ratio

Data for a Field and Timestep

$$\max_{\vec{c} \in U} Q\left(d_{f,t}, \widetilde{d_{f,t}}\left(\vec{c};\ \overrightarrow{\theta_c}\right)\right)$$

Error Bound

Allowed Error Bounds

FRaZ: Approach

Formulate compressor configuration as an optimization problem

Compression Ratio

Data for a Field and Timestep

$$\max_{\vec{c} \in U} Q\Big(d_{f,t}, \underbrace{\widetilde{d_{f,t}}\big(\vec{c};\ \overrightarrow{\theta_c}\big)}_{\text{Compression Function}}\Big)$$

Error Bound

Allowed Error Bounds

FRaZ: Approach

Formulate compressor configuration as an optimization problem

Compression Ratio

Data for a Field and Timestep

Error Bound Mode

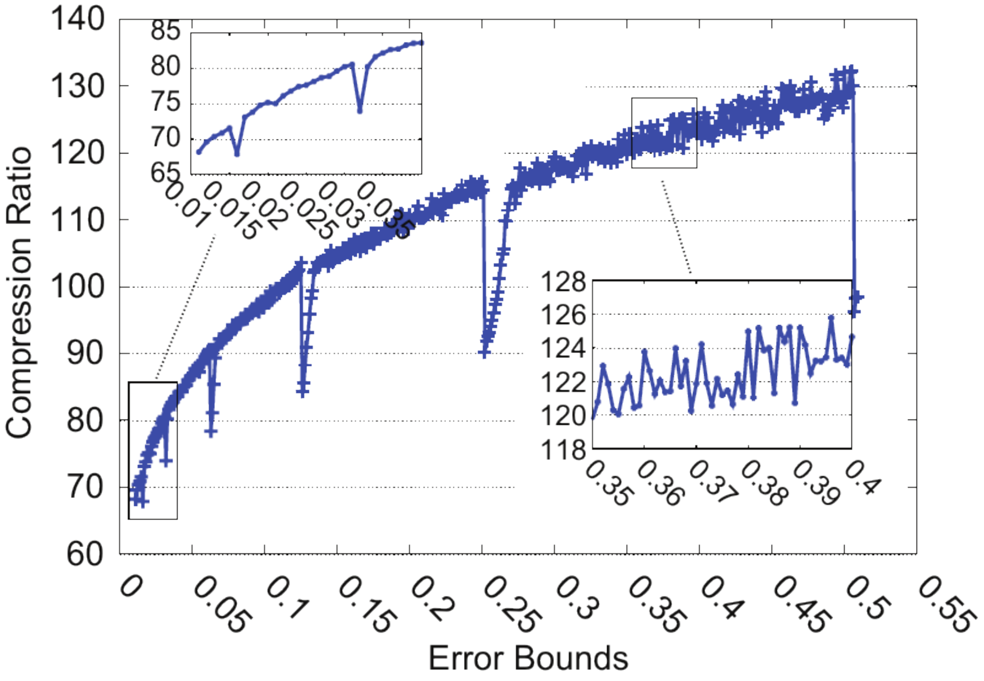
$$\max_{\vec{c}\in U} Q\left(d_{f,t}, \underbrace{\widetilde{d_{f,t}}\left(\vec{c};\ \overrightarrow{\theta_c}\right)}_{\text{Compression Function}}\right)$$

Error Bound

Allowed Error Bounds

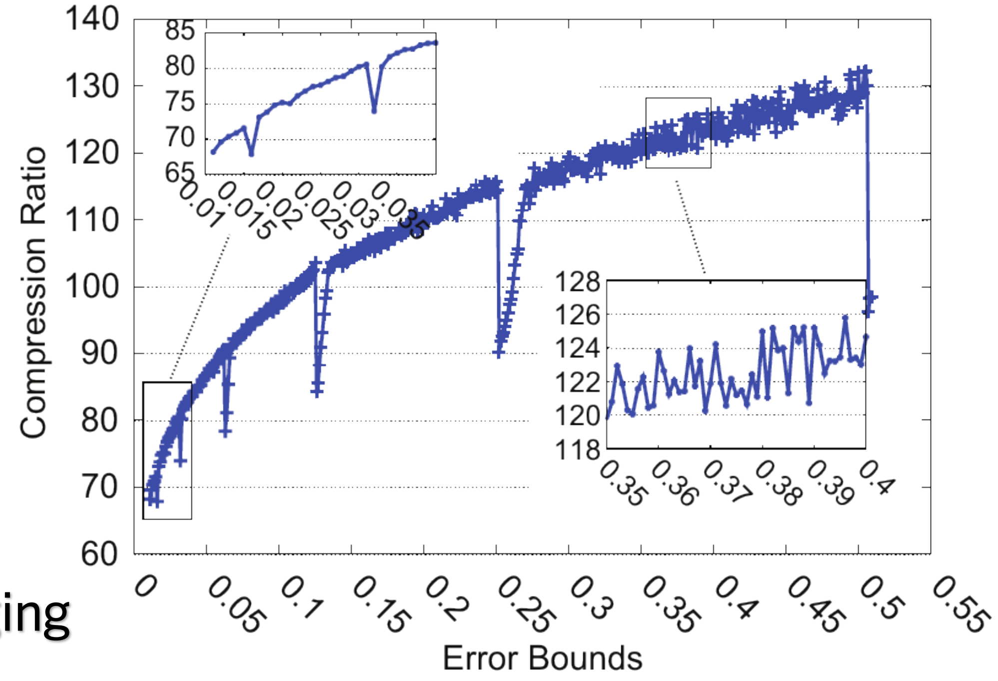
FRaZ: Approach

- Why not use binary search?
  - It doesn't work
  - The relationship between error bounds and compression ratios is not monotonic
- What can we use?
  - Optimization
    - Derivative Based Methods
      - Analytic Derivatives
      - Numerical Derivatives
    - Derivative Free Optimization



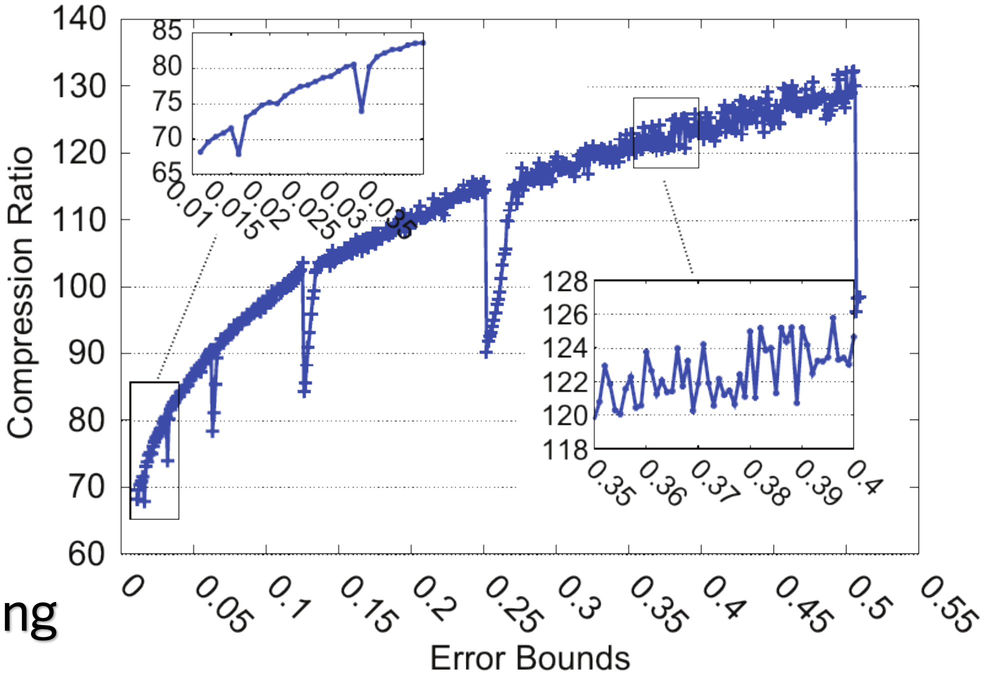
FRaZ: Approach

- Why not use binary search?
  - It doesn't work
  - The relationship between error bounds and compression ratios is not monotonic
- What can we use?
  - Optimization
    - Derivative Based Methods
      - ~~Analytic Derivatives~~ too challenging
      - Numerical Derivatives
    - Derivative Free Optimization



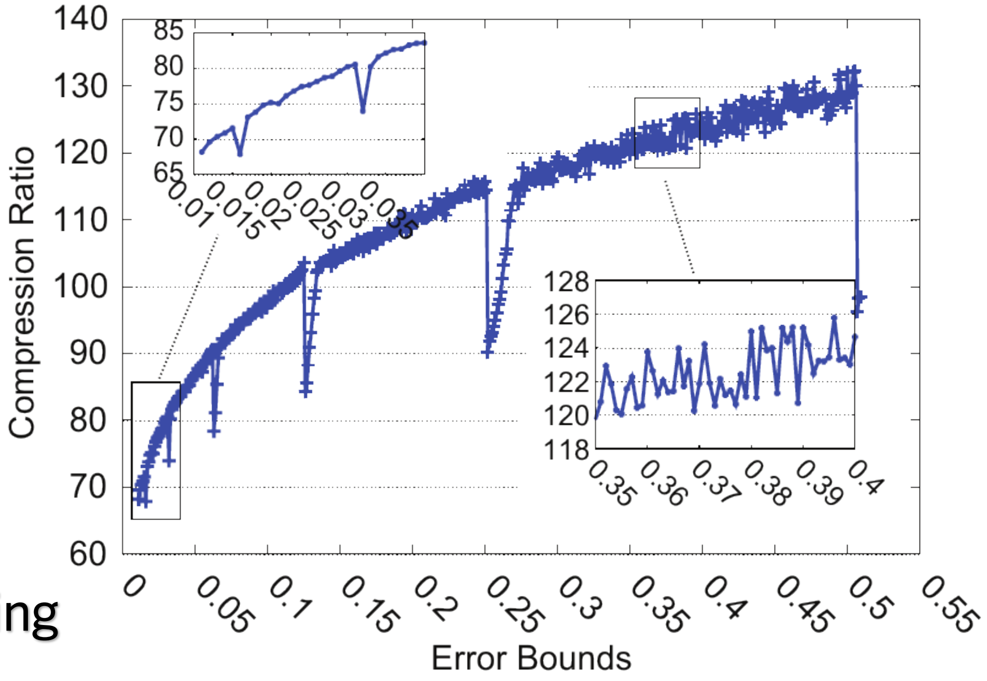
FRaZ: Approach

- Why not use binary search?
  - It doesn't work
  - The relationship between error bounds and compression ratios is not monotonic
- What can we use?
  - Optimization
    - Derivative Based Methods
      - ~~Analytic Derivatives~~ too challenging
      - ~~Numerical Derivatives~~ too slow
    - Derivative Free Optimization



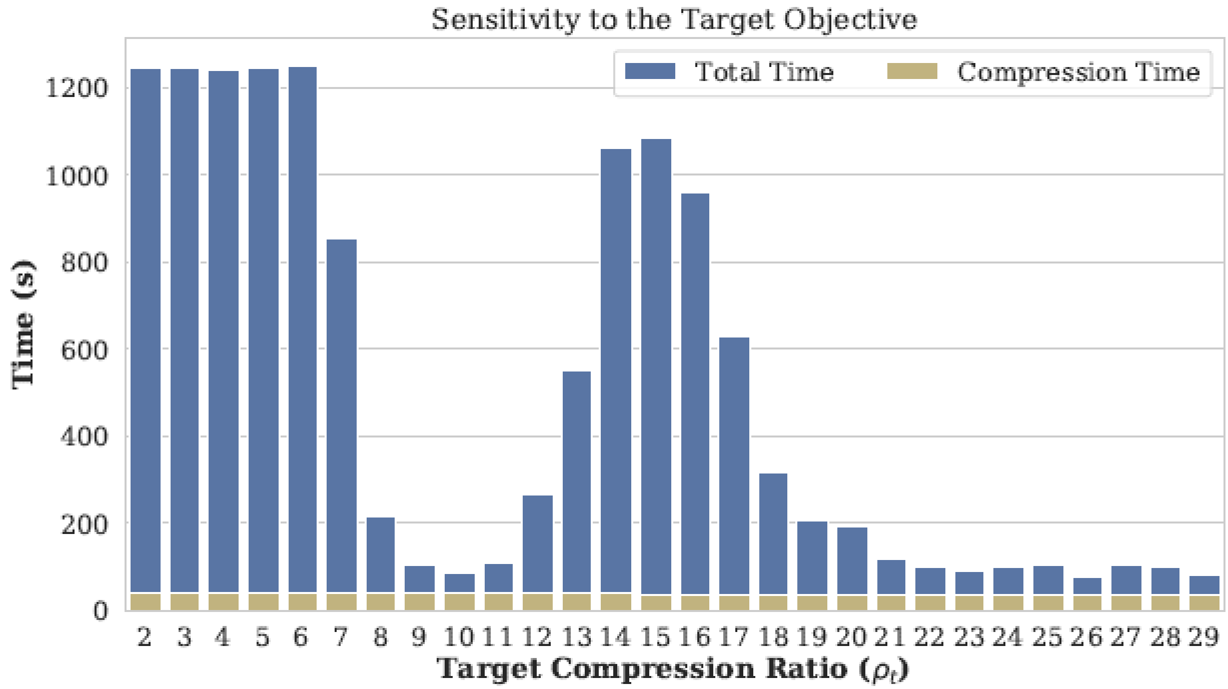
FRaZ: Approach

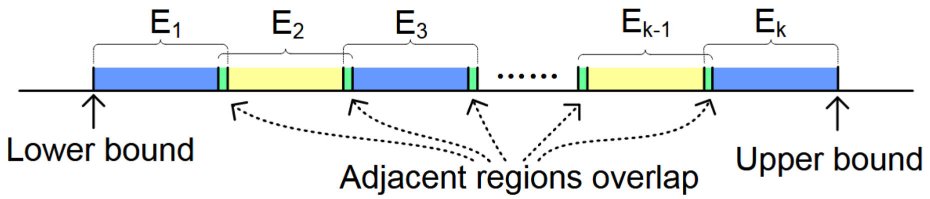
- Why not use binary search?
  - It doesn't work
  - The relationship between error bounds and compression ratios is not monotonic
- What can we use?
  - Optimization
    - Derivative Based Methods
      - ~~Analytic Derivatives~~ too challenging
      - ~~Numerical Derivatives~~ too slow
    - Derivative Free Optimization



FRaZ: Key Findings

- Tuning takes only 2x longer than an oracle in the feasible case
- Some targets are faster because more error bounds meet some targets
- How do we get there?
  - Parallelize by
    1. Field – run each field independently
    2. Timestep – try reusing prior timesteps configuration
    3. Error Bound Range – run ranges independently, stopping early if a solution is found





Automated Configuration Timeline

In Submission

↓

FRaZ
- Bounding Compression Ratio

→

LibPressio-Opt
- Bounding User Metrics
- Performance Improvements

→

More Complex Apps
- Bounding Complex, Multi-faceted Metrics

2 – LibPressio-Opt: Bound User Metrics

Can we extend FRaZ to bound simple user metrics and improve performance?

LibPressio-Opt: Approach

Formulate compressor configuration as an optimization problem

Compression Ratio

Data for a Field and Timestep

Error Bound Mode

$$\max_{\vec{c}\in U} Q\left(d_{f,t}, \underbrace{\widetilde{d_{f,t}}\left(\vec{c};\ \overrightarrow{\theta_c}\right)}_{\text{Compression Function}}\right)$$

Error Bound

Allowed Error Bounds

LibPressio-Opt: Approach

Formulate compressor configuration as an optimization problem

Compression Ratio

Data for a Field and Timestep

Error Bound Mode

$$\max_{\vec{c}\in U} Q\left(d_{f,t}, \underbrace{\widetilde{d_{f,t}}\left(\vec{c};\ \overrightarrow{\theta_c}\right)}_{\text{Compression Function}}; \overrightarrow{\boldsymbol{\theta}_{\boldsymbol{m}}}\right)$$

Error Bound

Allowed Error Bounds

LibPressio-Opt: Approach

Formulate compressor configuration as an optimization problem

Compression Ratio

Data for a Field and Timestep

Error Bound Mode

$$\max_{\vec{c}\in U} Q\left(d_{f,t}, \underbrace{\widetilde{d_{f,t}}\left(\vec{c};\ \overrightarrow{\theta_c}\right)}_{\text{Compression Function}}; \overrightarrow{\theta_m}\right)$$

Fixed Metric Parameters

Error Bound

Allowed Error Bounds

LibPressio-Opt: Approach

Formulate compressor configuration as an optimization problem

User Error Bound

Data for a Field and Timestep

Error Bound Mode

$$\max_{\vec{c}\in U} Q\left(d_{f,t}, \underbrace{\widetilde{d_{f,t}}\left(\vec{c};\ \overrightarrow{\theta_c}\right)}_{\text{Compression Function}}; \overrightarrow{\theta_m}\right)$$

Fixed Metric Parameters

Error Bound

Allowed Error Bounds

LibPressio-Opt: Approach

Formulate compressor configuration as an optimization problem

User Error Bound

Data for a Field and Timestep

Error Bound Mode

$$\max_{\vec{c}\in U} Q\Big(d_{f,t}, \underbrace{\widetilde{d_{f,t}}\big(\vec{c};\ \overrightarrow{\theta_c}\big)}_{\text{Compression Function}}; \overrightarrow{\theta_m}\Big)$$

Fixed Metric Parameters

Non-fixed Compressor Settings

Allowed Error Bounds

LibPressio-Opt: Approach

Formulate compressor configuration as an optimization problem

User Error Bound

Data for a Field and Timestep

Error Bound Mode

$$\max_{\vec{c}\in U} Q\left(d_{f,t}, \widetilde{d_{f,t}}\left(\vec{c};\ \overrightarrow{\theta_c}\right); \overrightarrow{\theta_m}\right)$$

Compression Function

Fixed Metric Parameters

Non-fixed Compressor Settings

Feasible Compressor Settings

LibPressio-Opt: Approach

Formulate compressor configuration as an optimization problem

User Error Bound

Data for a Field and Timestep

Compressor Fixed-Parameters

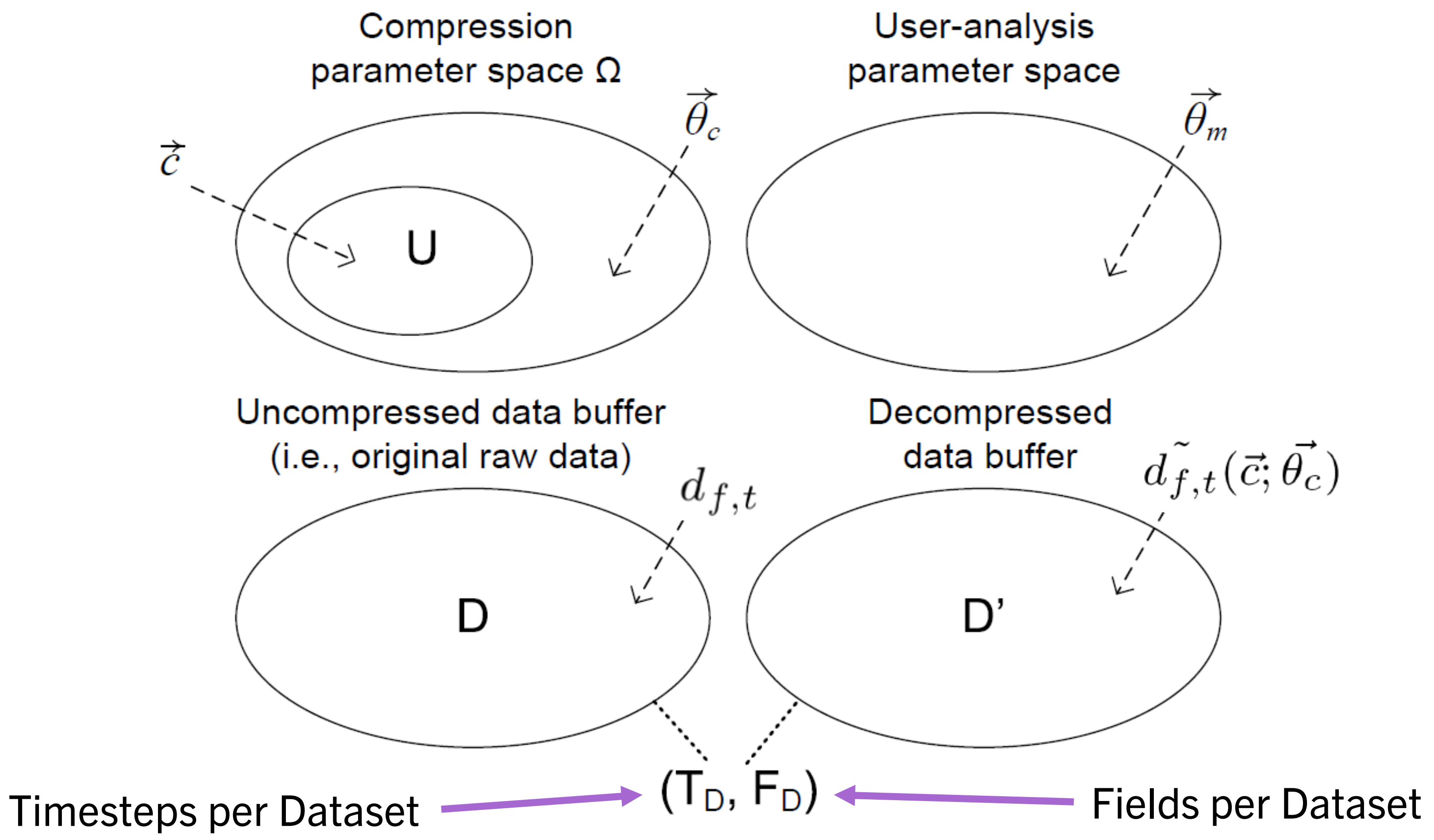
$$\max_{\vec{c}\in U} Q\left(d_{f,t}, \underbrace{\widetilde{d_{f,t}}\left(\vec{c};\ \overrightarrow{\theta_c}\right)}_{\text{Compression Function}}; \overrightarrow{\theta_m}\right)$$

Fixed Metric Parameters

Non-fixed Compressor Settings

Feasible Compressor Settings

Illustration of Relationship among Notations



What about constraints on objectives?

For any, $\underbrace{Q_i\left(d_{f,t}, \widetilde{d_{f,t}}(\vec{c};\ \overrightarrow{\theta_c})\ \right)}_{i^{th}\text{ metric}}$ and τ_i

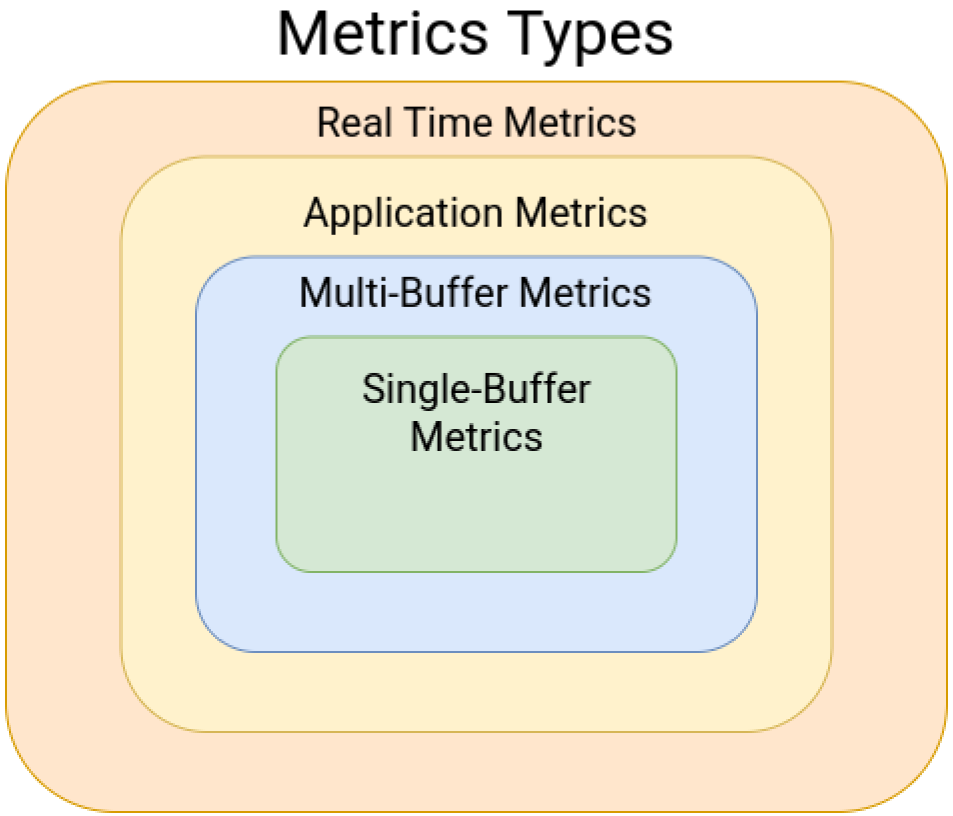
threshold for ith metric

We can construct:

$$Q\left(d_{f,t}, \widetilde{d_{f,t}}(\vec{c};\ \overrightarrow{\theta_c})\right) = \begin{cases} Q_0\left(d_{f,t}, \widetilde{d_{f,t}}(\vec{c};\ \overrightarrow{\theta_c})\right) \ if\ \forall_i, Q_i\left(d_{f,t}, \widetilde{d_{f,t}}(\vec{c};\ \overrightarrow{\theta_c})\right) \leq \tau_i \\ -\infty\ , otherwise \end{cases}$$

What do we mean by "Metrics"?

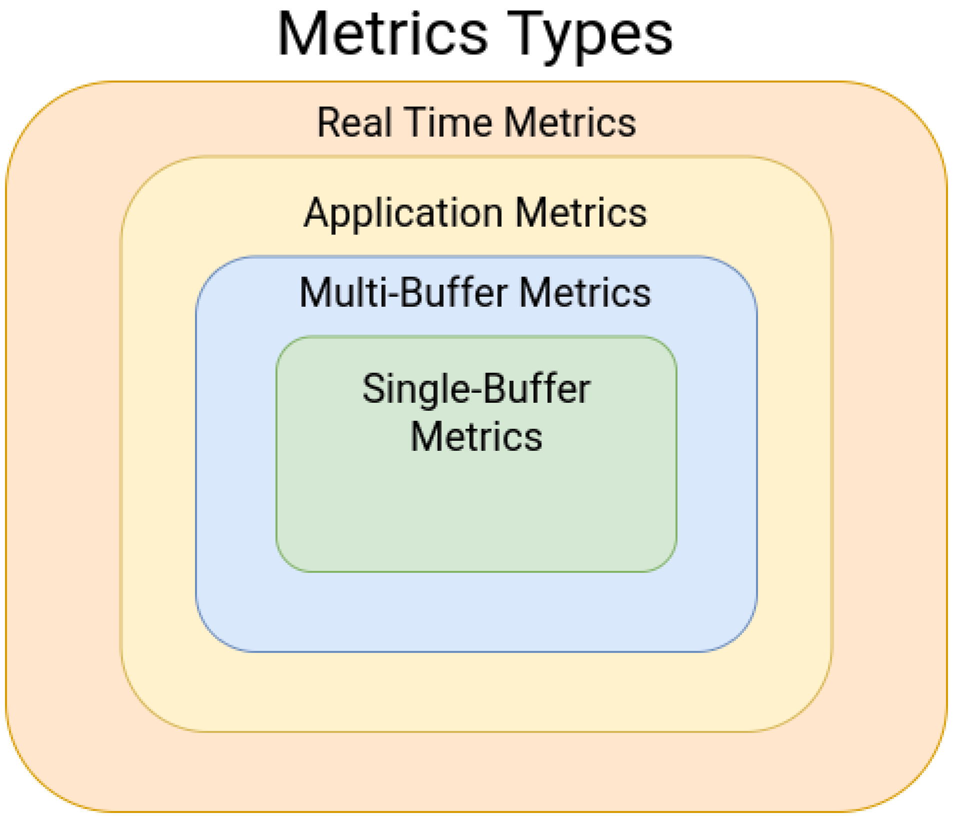
- Requirements
  - "Sufficiently" Deterministic
  - Have a fixed number of Real valued inputs and outputs
  - Can be modeled as having a single objective



What do we mean by “Metrics”?

- Requirements
  - “Sufficiently” Deterministic
  - Have a fixed number of Real valued inputs and outputs
  - Can be modeled as having a single objective
- Types

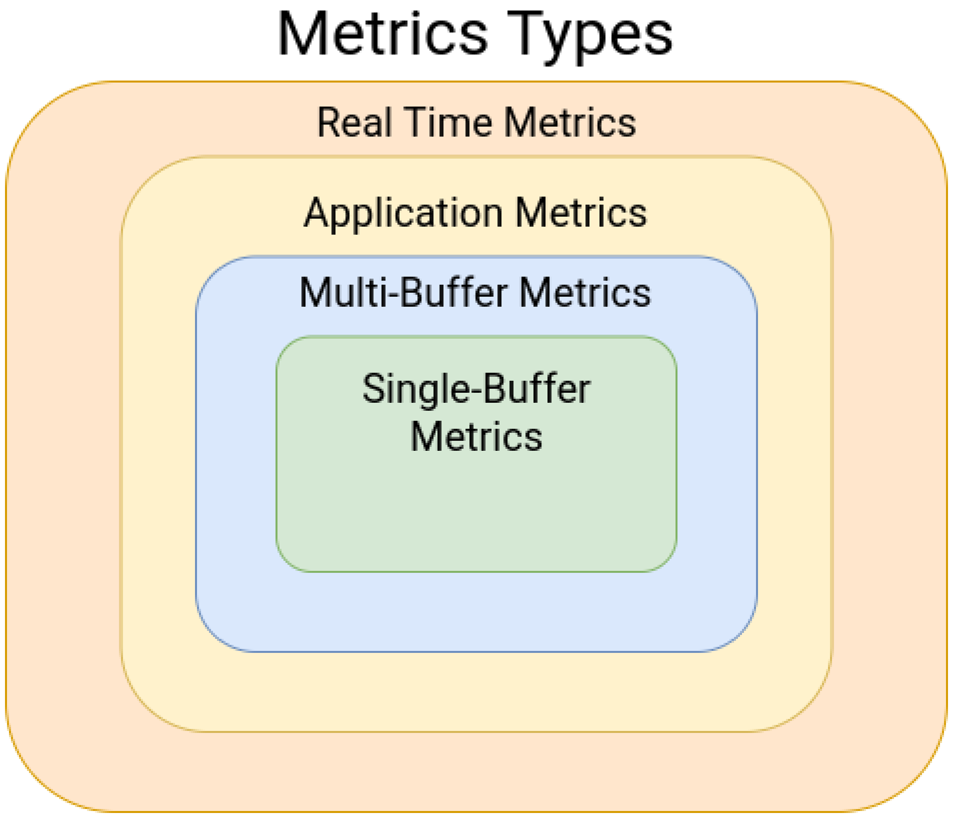
Real Time – Can only be computed at runtime (i.e. compression_time)



What do we mean by "Metrics"?

- Requirements
  - "Sufficiently" Deterministic
  - Have a fixed number of Real valued inputs and outputs
  - Can be modeled as having a single objective
- Types

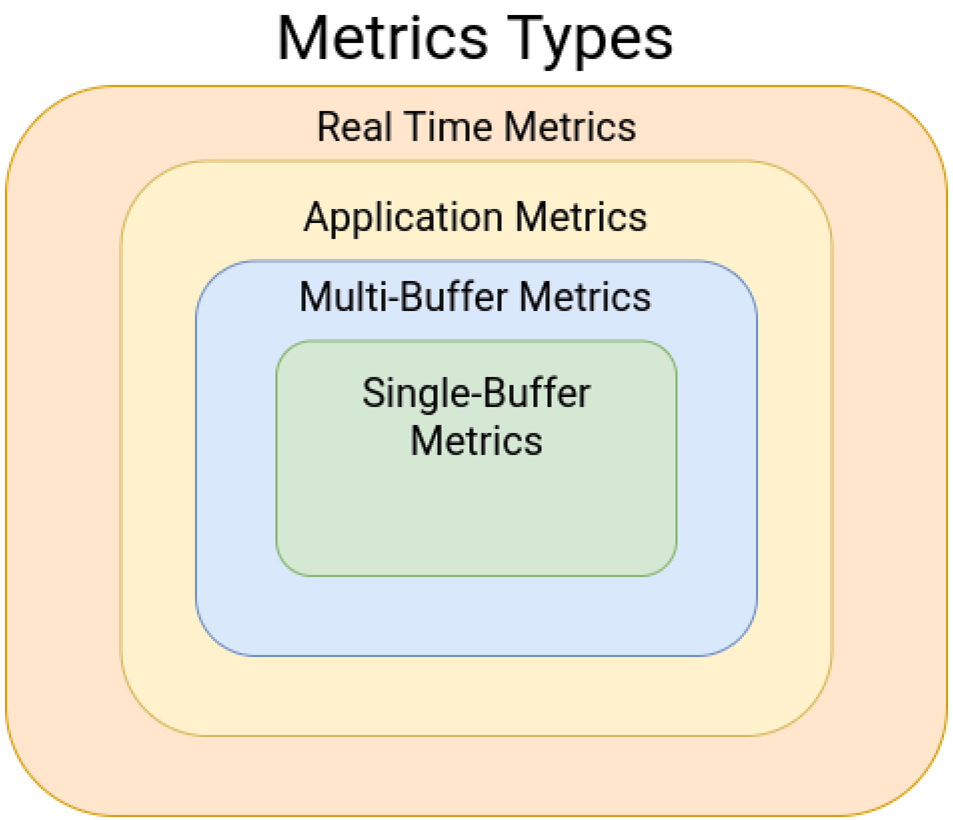
Application – Needs a specific collection of buffers to compute (Anything "app" specific)



What do we mean by "Metrics"?

- Requirements
  - "Sufficiently" Deterministic
  - Have a fixed number of Real valued inputs and outputs
  - Can be modeled as having a single objective
- Types

Multi-Buffer – Can be computed with any number of buffers (i.e. compression_ratio)

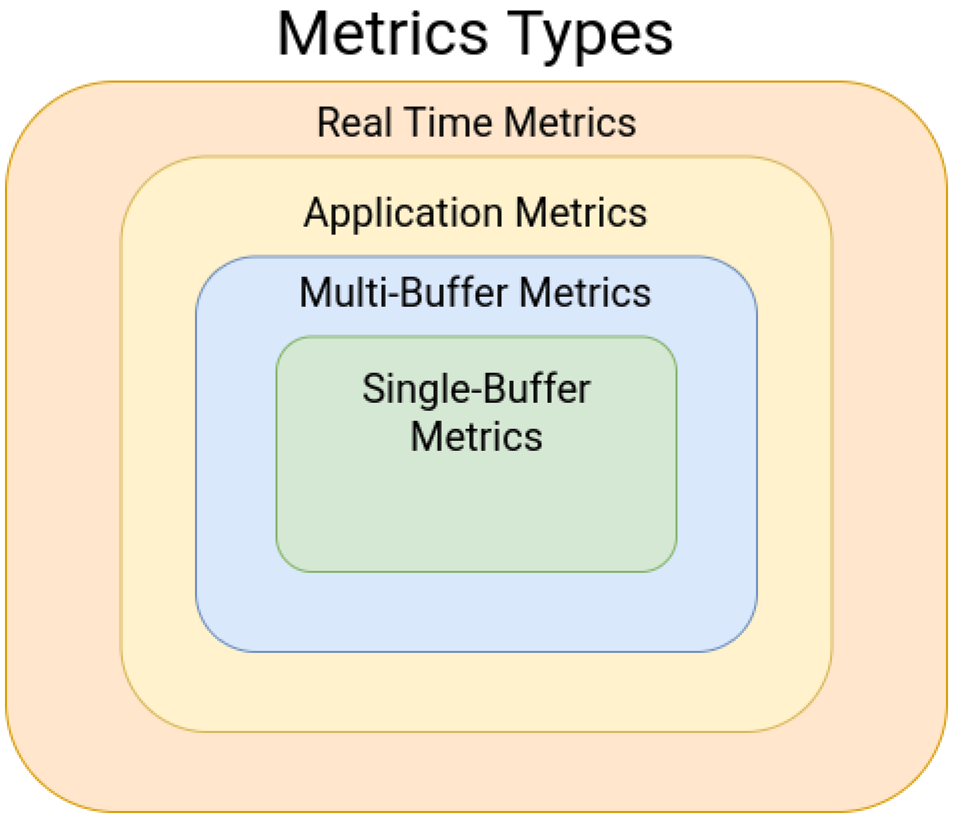


What do we mean by “Metrics”?

- Requirements
  - “Sufficiently” Deterministic
  - Have a fixed number of Real valued inputs and outputs
  - Can be modeled as having a single objective
- Types

Multi-Buffer – Can be computed with any number of buffers (i.e. compression_ratio)

Single-Buffer – Computed from any single buffer

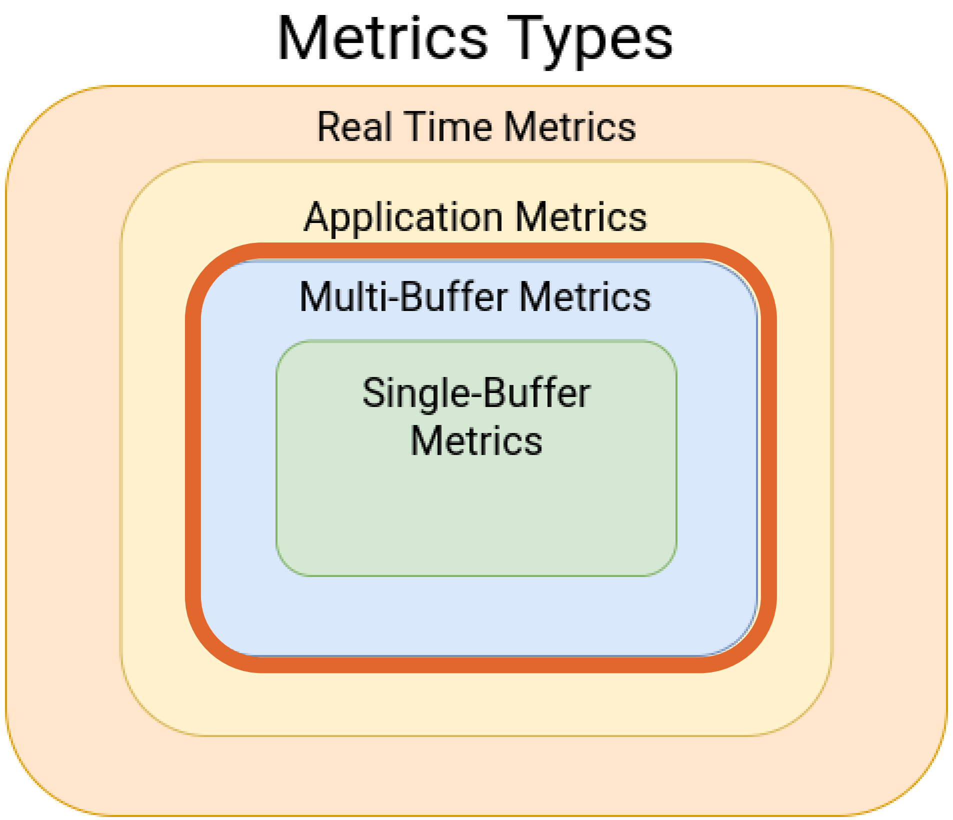


What do we mean by "Metrics"?

- Requirements
  - "Sufficiently" Deterministic
  - Have a fixed number of Real valued inputs and outputs
  - Can be modeled as having a single objective
- Types

Multi-Buffer – Can be computed with any number of buffers (i.e. compression_ratio)

Single-Buffer – Computed from any single buffer



MGARD Quantity of Interest Mode

- Requirements
  - Q is a bounded linear functional
    - iff: $Q(\alpha x + \beta y) = \alpha Q(x) + \beta Q(y)$
  - $d_{f,t}$ represents a regular grid
    - This includes many simulations
- Procedure
  - Precompute scaling factor $\Upsilon_{L^s}(Q)$
  - Use bound $\Upsilon_{L^s}(Q) \left\| d_{f,t} - \widetilde{d_{f,t}} \right\|_{L^s}$
  - Details in the paper cited below

Ainsworth, Mark; Tugluk, Ozan; Whitney, Ben; Klasky, Scott . "Multilevel techniques for compression and reduction of scientific data-quantitative control of accuracy in derived quantities. 2019

VS. MGARD Quantity of Interest Mode

- Relative to MGARD-QOI mode
  - LibPressio-Opt+SZ is much faster for one-off tasks
  - Even if precomputation is not required, it can still be faster

VS. FRaZ

- Relative to FRaZ
  - Inter-iteration early termination
  - Multi-threaded searches
  - Embeddable
  - Supports user-defined objectives
  - Supports multiple input parameters
  - Extendable Search methods

Automated Configuration Timeline

Current Work

FRaZ

- Bounding Compression Ratio

LibPressio-Opt

- Bounding User Metrics
- Performance Improvements

More Complex Apps

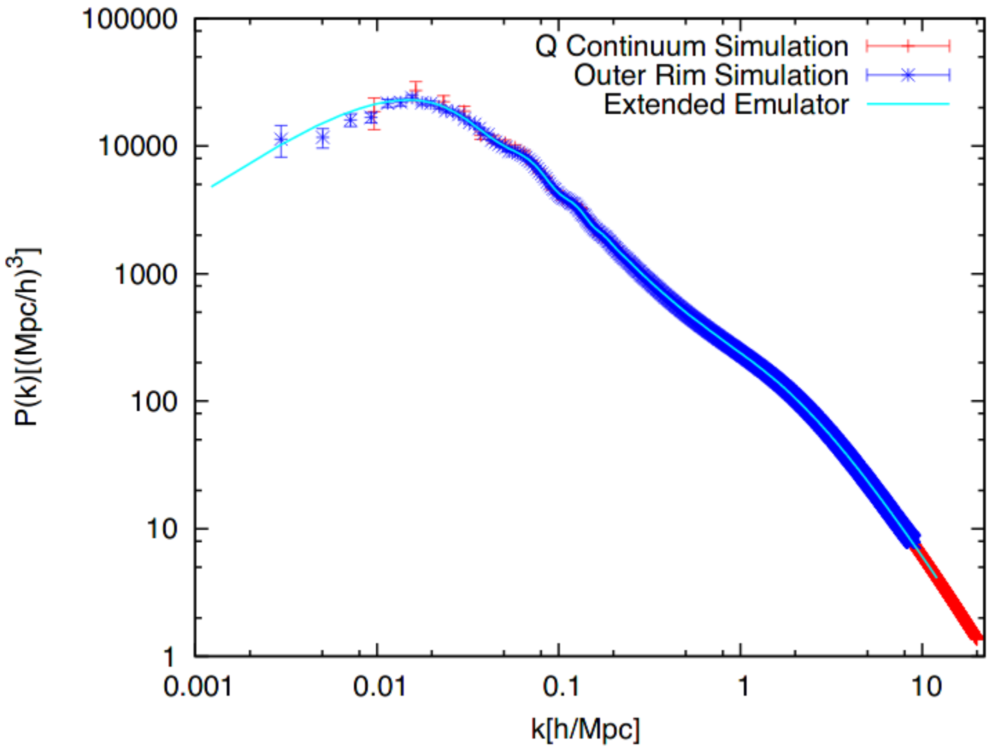
- Bounding Complex, Multi-faceted Metrics

3 – More Complex Applications

Can we adapt LibPressio-Opt to bound complex , multi-faceted metrics that use multiple buffers such as Hardware/Hybrid Accelerated Cosmology Code (HACC)'s power spectrum?

Background

- HACC – ECP astrophysics particle application
- No compressors bound spectral error



Matter Fluctuation Power Spectra from Outer Rim and Q Continuum simulation at z=0.26. [3]

Approach

1. Implement the spectra as a LibPressio metric of type vector<double>
2. Explore metrics to compare spectra
3. Solve maximum compression ratio such that differences between spectra are acceptable

Future Work

- True Multi-Objective Compression
- Improve the search algorithm
- Better Compression task scheduling

Understanding the Effects of Compression on ML/AI

Lossy Compression for AI

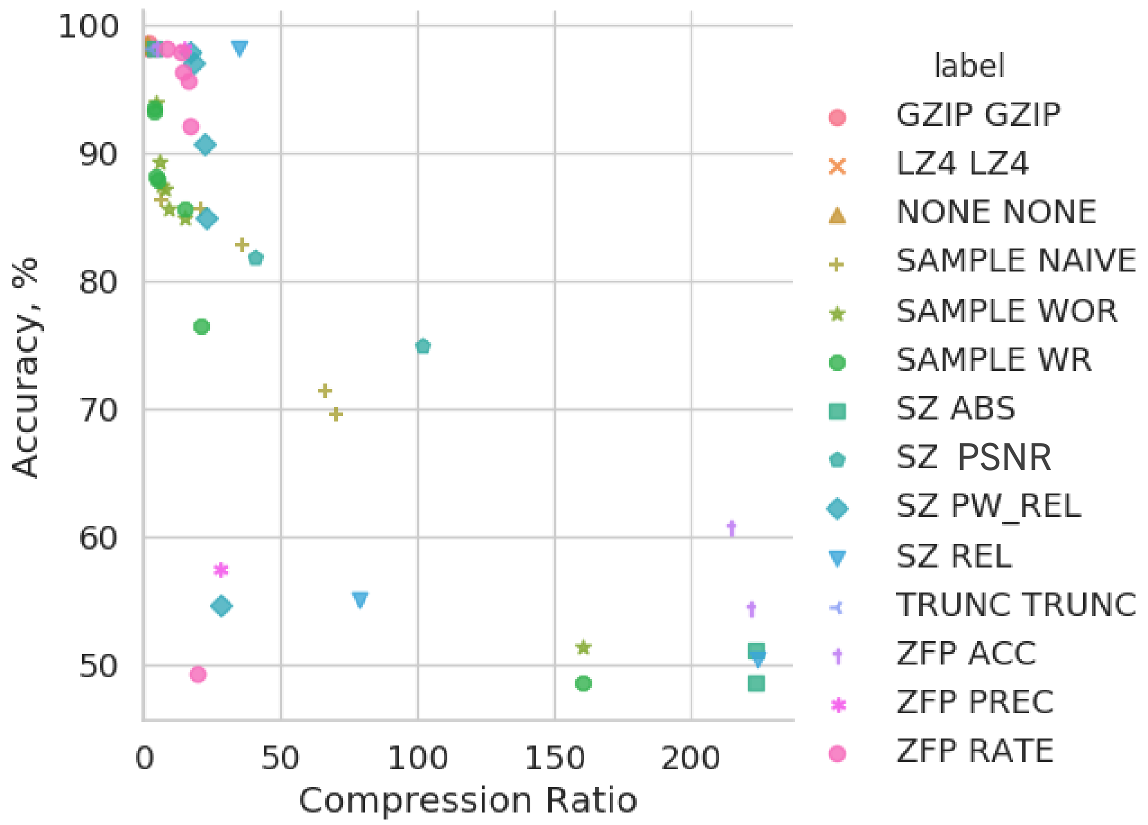
What are the trade-offs for compressing training and testing data to save storage space?

Approach

- Use LibPressio External Metrics on training data to collect pareto-optimal points
  - External Metrics run scripts to collect data.
  - In this case: here a known-good AI based model

Key Findings

- Prediction-Based EBLC works best (SZ)
  - Even better than sampling!
  - Even on imbalanced datasets!
- Sometimes EBLC improves performance!
- Compress tabular data relatively by feature



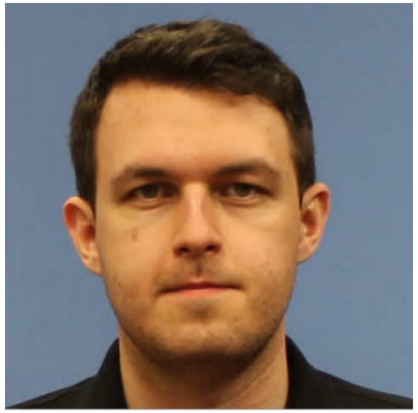
Conclusion

- Error Bounded Lossy Compression has the potential to be transformative
  - Especially with an interface to unify, tools to configure, and tools to understand

Thank You!










Thank You!

63

Questions?

Approachable Error Bounded Lossy Compression

Robert Underwood
robertu@g.clemson.edu
Clemson University
October 7, 2020

My Curriculum Vita

Papers I Worked On

- R. Underwood, S. Di, J. C. Calhoun and F. Cappello, "FRaZ: A Generic High-Fidelity Fixed-Ratio Lossy Compression Framework for Scientific Floating-point Data," 2020 (IPDPS)
- Jiannan Tian et al. “cuSZ: An Efficient GPU Based Error-Bounded Lossy Compression Framework for Scientific Data”. In: Proceedings of 29th International Conference on Parallel Architectures and Compilation Techniques. Co-Author. ACM. Atlanta, Georgia (virtual), Oct. 2020.
- A. Gok et al. “Metrics for the Preservation of the Error In Derivatives” 2020 (In Preparation)
- R. Underwood, S. Di, J. C. Calhoun and F. Cappello, “LibPressio-Opt: Fast User Error Bounds for Loss Compression" 2020 (In Preparation)
- R. Underwood et al. “Machine Learning and AI with Error Bounded Lossy Compression” (In Preparation)

References

- Ainsworth, Mark; Tugluk, Ozan; Whitney, Ben; Klasky, Scott . "Multilevel techniques for compression and reduction of scientific data-quantitative control of accuracy in derived quantities. 2019
- Di, Sheng and Cappello, Franck “Fast Error Bounded Lossy Compression with SZ” 2016
- Franck Cappello et al. “Use Cases of Lossy Compression for Floating Point Data in Scientific Datasets. 2018
- Habib, Salman, et al. "HACC: Simulating sky surveys on state-of-the-art supercomputing architectures." New Astronomy 42 (2016): 49-65.
- Lindstrom, Peter. “Fixed Rate Compressed Floating Point Arrays” 2012
- Whitney, Ben E. “Multilevel Techniques for Compression and Reduction of Scientific Data” 2018

- Machine Characteristics from respective websites accessed 17 September 2020
- Logos are the property of the respective institutions

Funding Acknowledgements

Thank you to Clemson Computing and Information Technology for funding my participation in SC20

This research was supported by the Exascale Computing Project (ECP),Project Number: 17-SC-20-SC, a collaborative effort of two DOE organizations - the Office of Science and the National Nuclear Security Administration, Responsible for the planning and preparation of a capable Exascale ecosystem, including software, applications, hardware, advanced system engineering and early testbed platforms, to support the nation's Exascale computing imperative. The material was supported by the U.S. Department of Energy, Office of Science, under contract DE-AC02-06CH11357, and supported by the National Science Foundation under Grant No. 1619253 and 1910197.We acknowledge the computing resources provided on Bebop, which is operated by the Laboratory Computing Resource Center at Argonne National Laboratory. This material is also based upon work supported by the U.S. Department of Energy, Office of Science, Office of Workforce Development for Teachers and Scientists, Office of Science Graduate Student Research (SCGSR) program. The SCGSR program is administered by the Oak Ridge Institute for Science and Education (ORISE) for the DOE. ORISE is managed by ORAU under contract number DE-SC0014664. All opinions expressed in this paper are the authors and do not necessarily reflect the policies and views of DOE, ORAU, or ORISE

Installing LibPressio

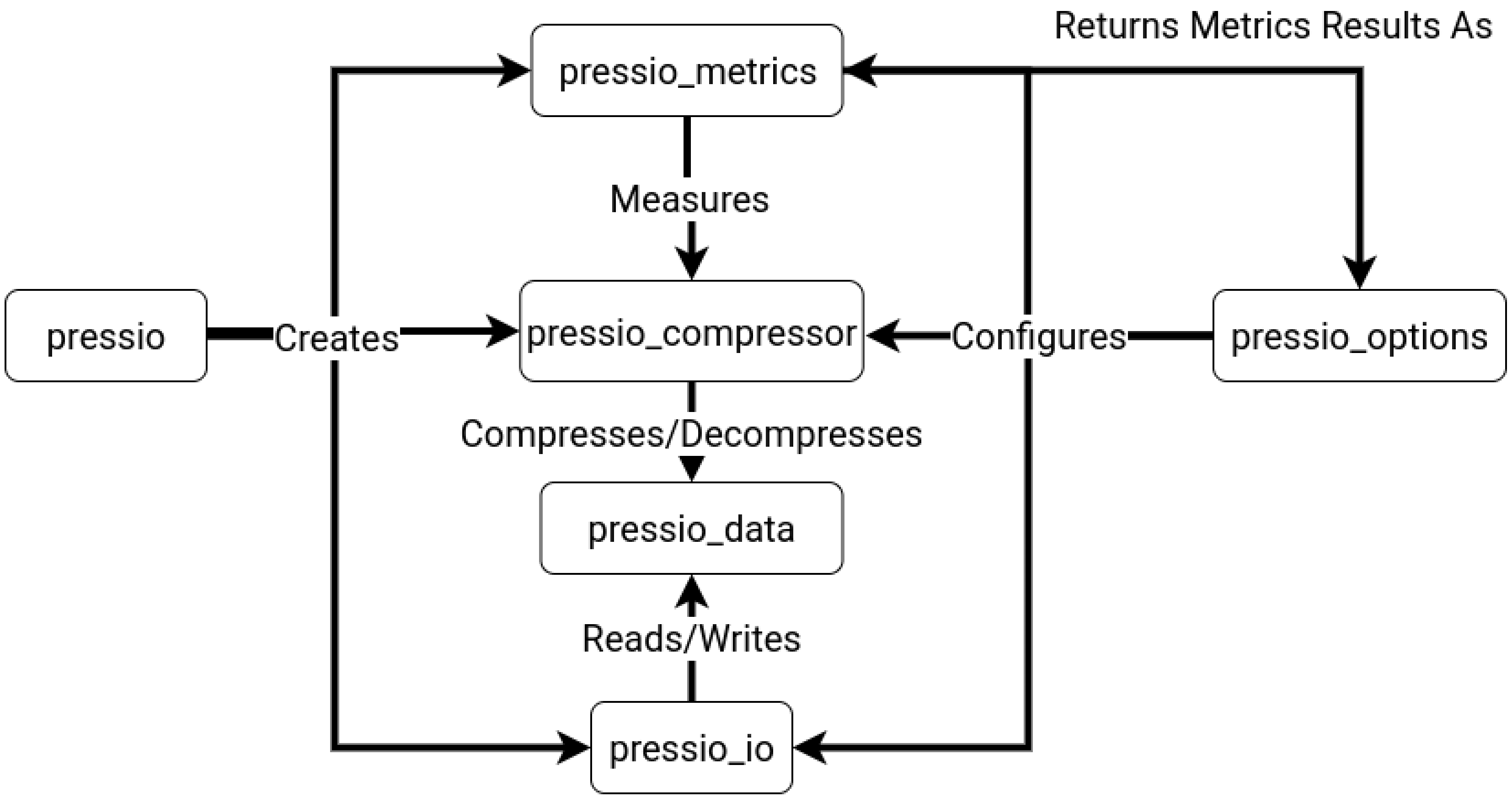
Prerequisites:

- LibPressio is currently only supported on Linux
- python>=3.4, tar, bzip, xz, gzip, clang/gcc, curl, git, bash, patch

Installation:

```
git clone https://github.com/spack/spack
git clone https://github.com/robertu94/spack_packages robertu94_packages
source ./spack/share/spack/setup-env.sh
spack compiler find
spack repo add ./robertu94_packages
spack install libpressio-tools ^libpressio+sz+zfp+lua
spack load libpressio-tools
```

Overview of the structures in LibPressio



pressio – library control

- Query
  - Supported compressors
  - Version information
- Create compressor instances

```
//get the compressor
struct pressio* library = pressio_instance();
struct pressio_compressor* sz = pressio_get_compressor(library,
↪  "sz");
```

pressio_options – key-value store

- map<string, option>
- One structure multiple uses:
  - Options – runtime settings
  - Configuration – compile time settings
  - Metrics Results – runtime observations
- Introspection
- Get/Set/Cast
- Validation of Settings

```
//configure, validate, and assign the options
struct pressio_options* config =
↪  pressio_compressor_get_options(sz);
pressio_options_set_integer(config, "sz:error_bound_mode",
↪  REL);
pressio_options_set_double(config, "sz:rel_err_bound", 0.01);
pressio_compressor_set_options(sz, config);
```

pressio_io

- Required Interfaces
  - read/write
- Optional Interfaces
  - *configurable*
    - {get,set,check}_options
    - get_configuration
  - {read,write}_many

```
//read in an input buffer
size_t dims[] = {500,500,100};
struct pressio_data* description =
↪   pressio_data_new_empty(pressio_float_dtype, 3, dims);
struct pressio_data* input_data =
↪   pressio_io_data_path_read(description, "CLOUDf48.bin.f32");
```

pressio_data – generic reference to data

- span<T>
- Query
  - type
  - size
  - values
  - owning/non-owning
- Extensible

```
//create output buffers
struct pressio_data* compressed_data =
↪   pressio_data_new_empty(pressio_byte_dtype, 0, NULL);
struct pressio_data* decompressed_data =
↪   pressio_data_new_owning(pressio_float_dtype, 3, dims);
```

pressio_compressor

- Required Interfaces:
  - compress/decompress
  - version
- Optional Interfaces:
  - {compress,decompress}_many
  - *configurable*
  - get_metrics_impl

```
//compress and decompress the data
pressio_compressor_compress(sz, input_data, compressed_data);
pressio_compressor_decompress(sz, compressed_data,
↪   decompressed_data);
```

pressio_metrics

- Required Interface
  - get_metrics_results
- Optional Interfaces
  - *configurable*
  - before_${compressor_fn}
  - after_${compressor_fn}

Using the Command Line

```
pressio \
  -t float -d 500 -d 500 -d 100 \
  -i $datasets/CLOUDf48.bin.f32 \
  -m size -m time \
  -M all \
  -o sz:abs_err_bound=1e-6 \
  -o sz:error_bound_mode_str="ABS" \
  sz
```

```
#use the command line
# the data is 32-bit float, dims=500x500x100
# the data is a “brick-of-floats” here
# collect size and time metrics
# print all collected metrics
# set SZ’s abs error bound to 1e-6
# set SZ’s error bound mode to ABS
# use SZ as the “root” compressor
```

Learn More about LibPressio

- Want to get Started?
  - Watch the "LibPressio Tutorial" on YouTube
  - Reference the extensive developer documentation
  - Check it out on Github

Get LibPressio