FRaZ: A Generic High-Fidelity Fixed-Ratio Lossy Compression Framework for Scientific Floating-point Data

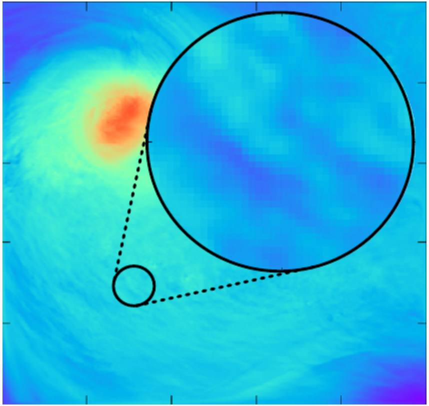
Robert Underwood †‡, Sheng Di ‡, Jon C. Calhoun †, and Franck Cappello ‡
Questions: robertu@g.clemson.edu
Video Presentation: https://youtu.be/oXpZAEEywHg
Code: https://www.github.com/CODARCode/FRaZ

May 18, 2020

Clemson University †, Argonne National Laboratory ‡

Why do we need Fixed Ratio Lossy Compression?

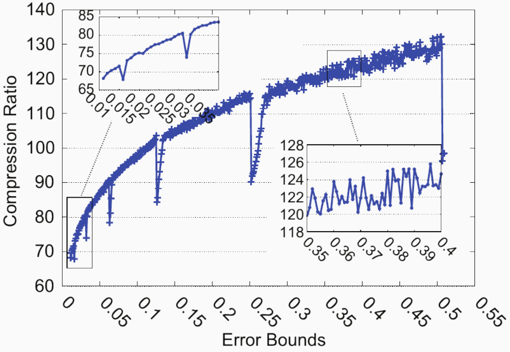
1. To reduce the storage footprint
    - The ORNL Summit limit: 50 TB/project
    - Many Scientific codes such as HACC or CESM produce 100s of TB if not PB of data
2. To achieve ”best fit” compression
    - Users want to store as they can in their available storage
    - Without fixed-ratio, they either suffer a loss in quality or result to trial and error
3. Streaming applications
    - Scientific instruments such as the APS and LCLS-II may generate image data rates exceeding 250GB/s.
    - However, the backing storage is limited to 25GB/s



Hurricane, dataset used in paper with zoom-in view

Why is this Difficult? Or Why Can't We Just Use Binary Search?

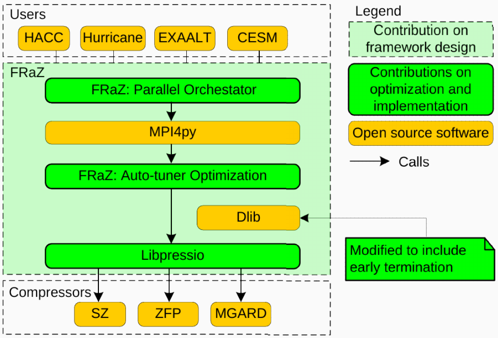
- Current compressors don't implement fixed-ratio compression or implement an similar "fixed-rate" mode which isn't error bounded (see paper)
- The relationship between error bound and compression ratio is not monotonic and non-convex for all compressors and datasets
- This is especially true of compressors like SZ which have a dictionary encoding stage
- White-box approaches (where the compressor is deeply known) quickly fall out of date



Non-monotonicity in the Hurricane dataset

Our Contributions are

- Formulated fixed-ratio compression as an optimization problem in a way that converges quickly
- Evaluated several different optimization algorithms to find one that works on all of our test cases, and then modified it to improve performance for our FRaZ
- Implemented and ran parallel search to improve the throughput of the technique



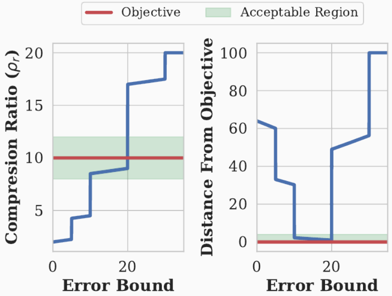
Overview of FRaZ Architecture and Contributions

Formulating Fixed Ratio Compression as an Optimization Problem

- **Given:**
  Original Dataset $D_{f,t}$
  Decompressed Dataset $D'_{f,t}$
  Fixed Compression Parameters θ
  Acceptable Compressor Error Bound U
  Real compression ratio $\rho_r(D_{f,t}, e, \theta)$
  Target compression ratio $\rho_t(D_{f,t})$
  Target compression ratio relative tolerance ϵ

Let: Compressor Error Bound e

- **Minimize over e:**
  $(\rho_r(D_{f,t}, e, \theta) - \rho_t(D_{f,t}))^2$ s.t. $0 \leq e \leq U$
  if $(\rho_r(D_{f,t}, e, \theta) - \rho_t(D_{f,t}))^2 \leq \epsilon^2 \rho_t(D_{f,t})$, terminate
- **Many Algorithms preform poorly:**
  We don't have a analytic forms for ρ_r, $\rho_r\prime$, or $\rho_r\prime\prime$
  Numerical derivatives are costly, $O(sec) - O(min)$
  Empirically, ρ_r often is non-convex many local optima

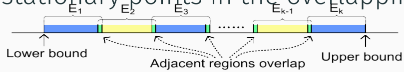


The Acceptable Region is where we can early terminate the search

- We choose Dlib's `find_global_min` – Lipschitz Optimization + NEWOUA, http://blog.dlib.net/2017/12/a-global-optimization-algorithm-worth.html

Parallelizing the Algorithm

1. By Field – embarrassingly parallel
2. By Timestep
   - Do first timestep as normal
   - Guess next solution is same as last
   - If wrong, do full tuning again
3. By Error Bound Range
   - Split search range $[0, U]$ into n similarly sized subranges run an independent search on each as hardware allows
   - a slight overlap (i.e. 10%) improves performance allowing for sufficient stationary points in the overlapping region



Algorithm 2 TRAINING

Input: target compression ratio $\rho_t(D_{f,t})$, acceptable error ϵ, dataset D_t, max allowed compression error U

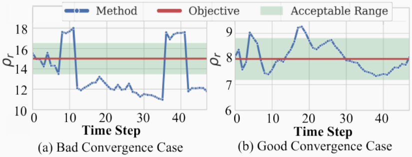
Output: real compression ratio $\rho_r(D_{f,t}, e)$, recommended error bound setting e

$tasks[N]$
$done \leftarrow false$
for $(i, (l, u)) \in make\_error\_bounds(U)$ **do**
    $tasks[i] \leftarrow launch\_task(D_t, l, u, \rho_t(D_{f,t}), \epsilon, h)$
end for
while $notdone$ **do**
    $last\_task \leftarrow next\_completed(tasks)$
    $candiate \leftarrow compression\_ratio(last\_task)$
    if $\rho_t(D_{f,t})(1 - \epsilon) \leq candidate \leq \rho_t(D_{f,t})(1 + \epsilon)$ **then**
        $done \leftarrow true$
        for $task \in tasks$ **do**
            $cancel\_if\_not\_finished(task)$
        end for
    end if
    $done \leftarrow has\_next(completed)$
end while
$\rho_r(D_{f,t}, e) = \infty$
for $task \in tasks$ **do**
    if finished(task) **then**
        $\rho \leftarrow compression\_ratio(task)$
        if $(\rho_r - \rho)^2 < (\rho_t - \rho)^2$ **then**
            $\rho_r = \rho$
        end if
    end if
end for
return $\rho_r(D_{f,t}, e), error\_bound(task)$

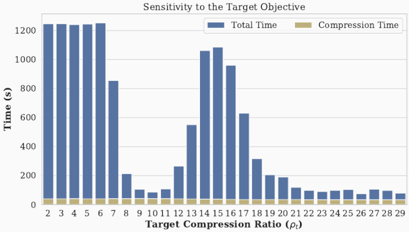
Worker Algorithm

Results: Time to Solution

- Runtime depends substantially if the requested target is feasible:
  - Good (feasible) Case: We terminate early most of the time
  - Bad (infeasible) Case: We alternate between a compression ratio which is too small or too large
- Very small compression ratios are often infeasible because there is a minimum compressed size
- There are also gaps between feasible and infeasible. For this figure $\rho_t(D_{f,t}) \in [14, 16]$ are infeasible for the specified ϵ
- In the feasible case, overhead is often $\approx$ 2x just compressing with the correct error bound.



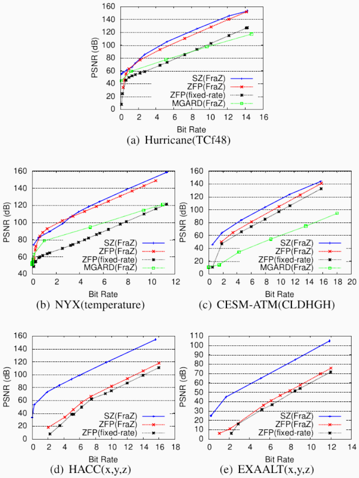
Solutions in good/bad case



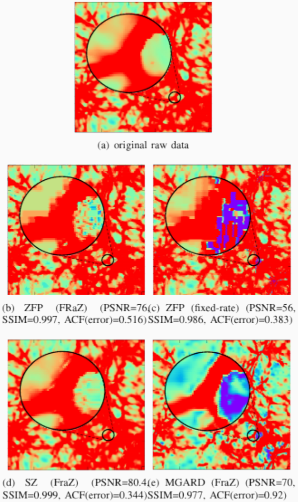
Time to solution for many targets

Results: Quality of Solution

- Fixed Ratio SZ/ZFP is generally better than ZFP Fixed Rate at each compression ratio:
    - Better Rate Distortion (higher PSNR per bit rate)
    - Higher SSIM
    - Higher PSNR
    - Better visual quality
- Figure 1: Rate Distortion for Several Datasets
- Figure 2: Visual Quality for Several Compressors



1



2

Conclusions

- Major Conclusions:
  - Fixed Ratio is better than existing Fixed Rate methods at preserving the data quality for equivalent compression ratios
  - Fixed Ratio Compression is higher performance when there are a large number of feasible compression ratios
  - We have relatively low overhead in the feasible case
- Future Work:
  - Arbitrary User Error Bounds – bounds that correspond with the quality of a scientist’s analysis result relative to that on noncompressed data
  - Online Version – Develop an online version of this algorithm to provide in situ fixed-ratio compression for simulation and instrument data.
  - Algorithm Improvements – Further improve the convergence rate of our algorithm to make it applicable for more use cases

Acknowledgements

This research was supported by the Exascale Computing Project (ECP), Project Number: 17-SC-20-SC, a collaborative effort of two DOE organiza- tions - 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 author's and do not necessarily reflect the policies and views of DOE, ORAU, or ORISE.