# Scaling Resiliency via machine learning and compression


**Alok Choudhary**

Henry and Isabel Dever Professor
EECS and Kellogg School of Management
Northwestern University
choudhar@eecs.northwestern.edu

Founder, chairman and Chief Scientist
4Cinsights Inc: A Big Data Science Company
+1 312 515 2562
alok@4Cinsights.com


U.S. DEPARTMENT OF ENERGY

National Science Foundation
WHERE DISCOVERIES BEGIN

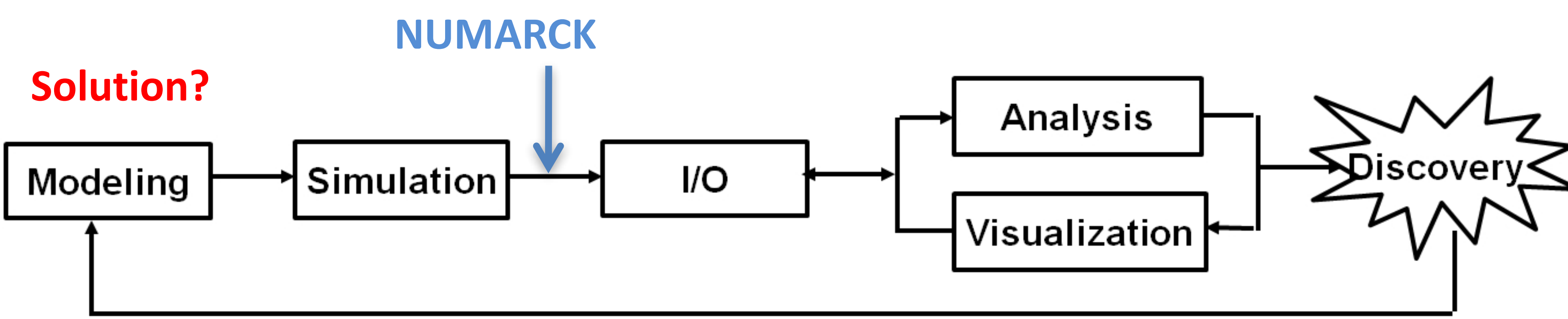
# Motivation

- Scientific simulations
  - Generate large amount of data.
  - Data feature: high-entropy, spatial-temporal
- Exascale Requirements*
  - Scalable System Software: Developing scalable system software that is power and resilience aware.
  - Resilience and correctness: Ensuring correct scientific computation in face of faults, reproducibility, and algorithm verification challenges.
- NUMARCK (NU Machine learning Algorithm for Resiliency and CheckPointing)
  - Learn temporal relative change and its distribution and bound point-wise user defined error.

* From *Advanced Scientific Computing Advisory Committee* Top Ten Technical Approaches for Exascale

# Checkpointing and NUMARACK

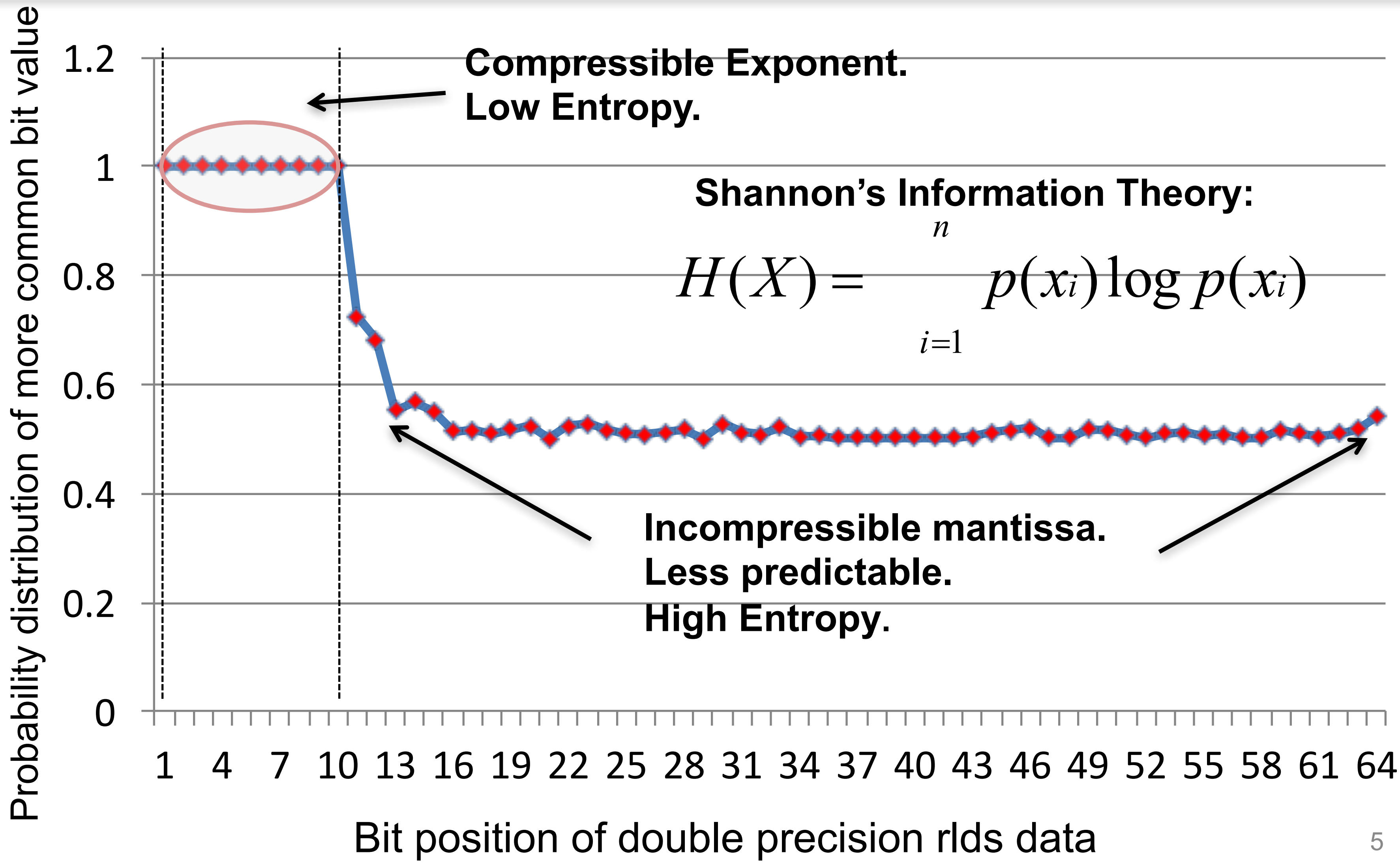
- Traditional checkpointing systems store raw (and uncompressed) data
  - **cost prohibitive: the storage space and time**
  - **threatens to overwhelm the simulation and the post-simulation data analysis**
- I/O accesses have become a limiting factor to key scientific discoveries.



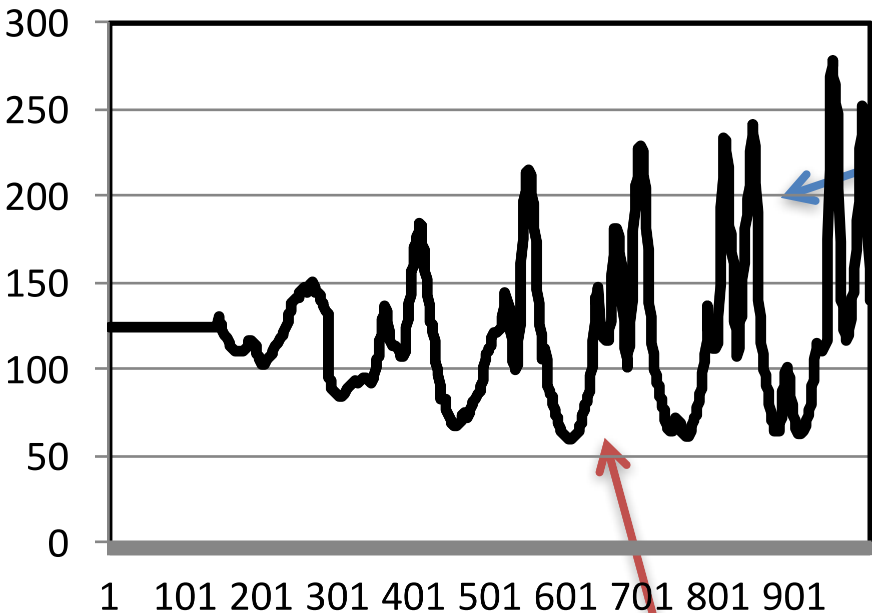
# What if a Resilience and Checkpointing Solution Provided

- Improved Resilience via more frequent yet relevant checkpoints, while
- Reducing the amount of data to be stored by an order of magnitude, and
- Guaranteeing **user-specified** tolerable maximum error rate for each data point, and
- an order of magnitude smaller mean error for each data set, and
- reduced I/O time by an order of magnitude, while
- Providing data for effective analysis and visualization

# **Motivation**: “Incompressible” with Lossless Encoding



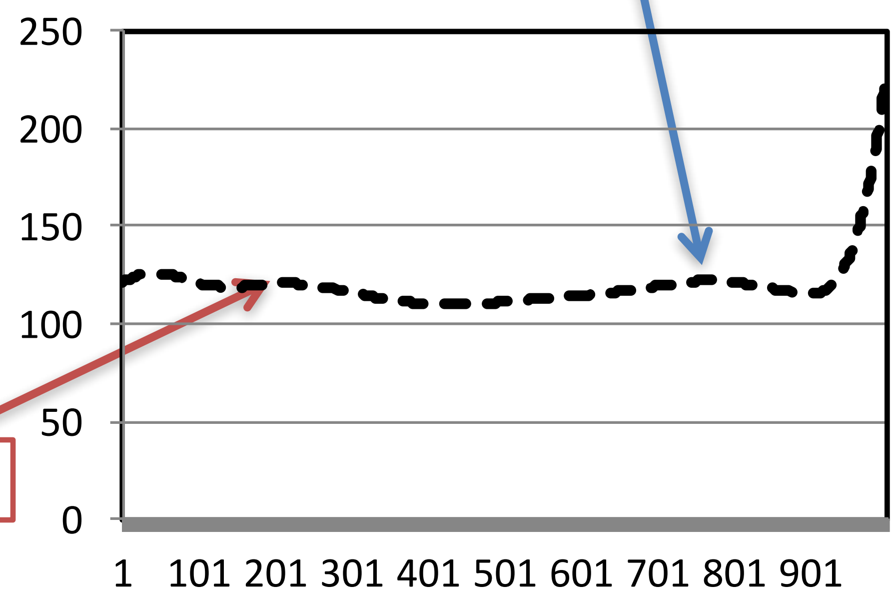
# **Motivation**: Still "Incompressible" with Lossy Encoding




Original rlds data

- **Highly random**
- **Extreme events missed**

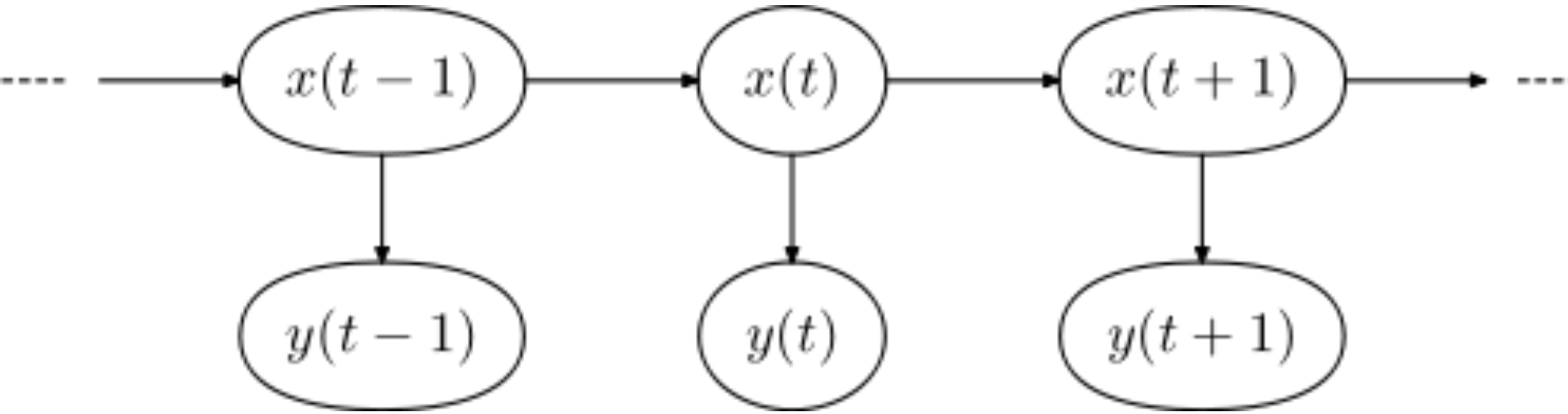
~0.35 correlation!




Bspline reconstructed rlds data

# Observation: Simulation Represents a State Transition Model

# What if we analyze the Change in Value?



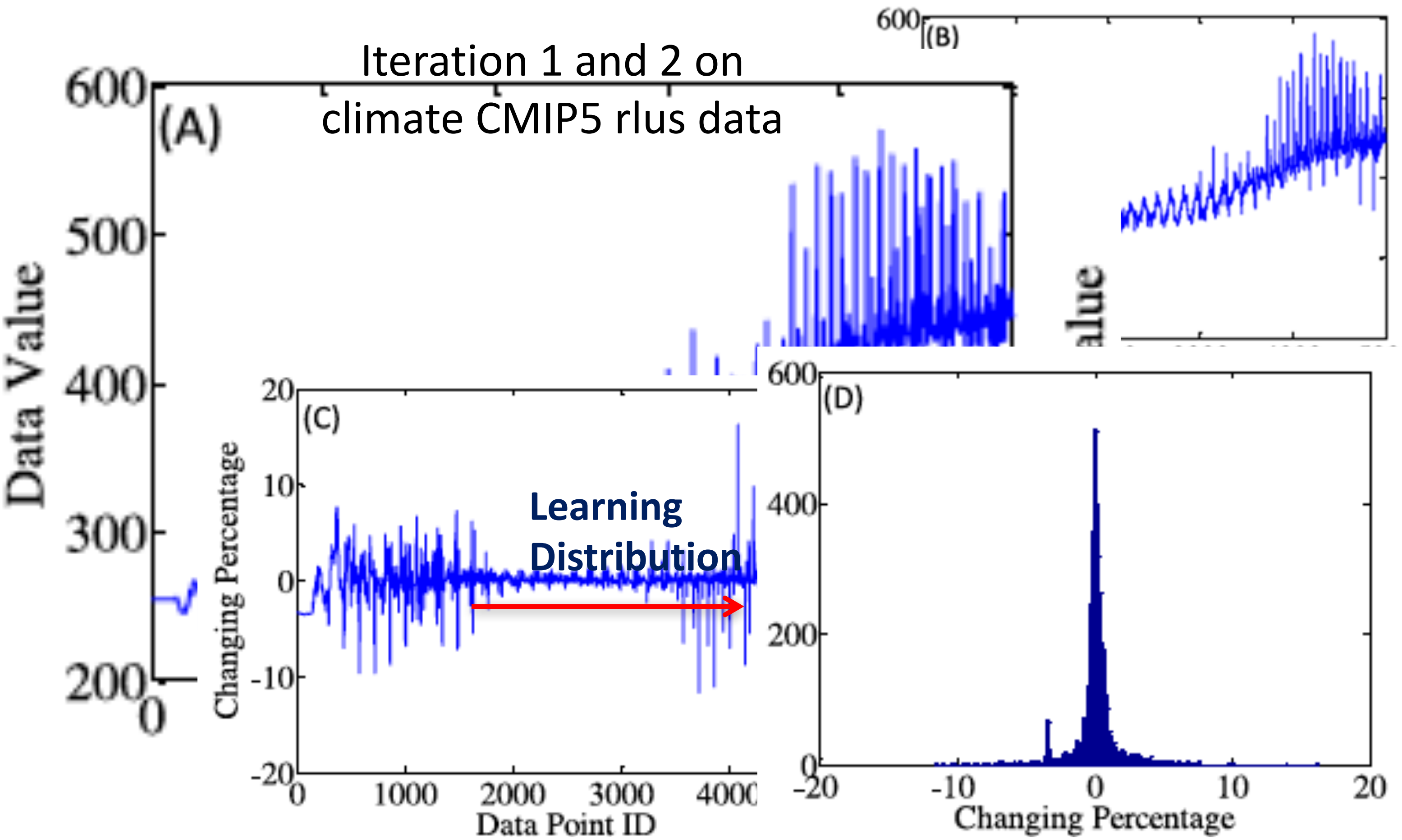

Observations:
- Variable Values – distribution
- Change in Variable Value – distribution
- Relative Change in Variable Value - distribution

$$\Delta D_{i,j} = \frac{D_{i,j} - D_{i-1,j}}{D_{i-1,j}}$$

Hypothesis: **The relative changes in variable values can be represented in a much smaller state space.**

- A1(t) = 100, A1(t+1) = 110 => change = 10, rel change = 10%
- A2(t) = 5, A2(t+1) = 5.5 => change = .5, rel change = 10%

# Sneak Preview: Relative Change is more predictable

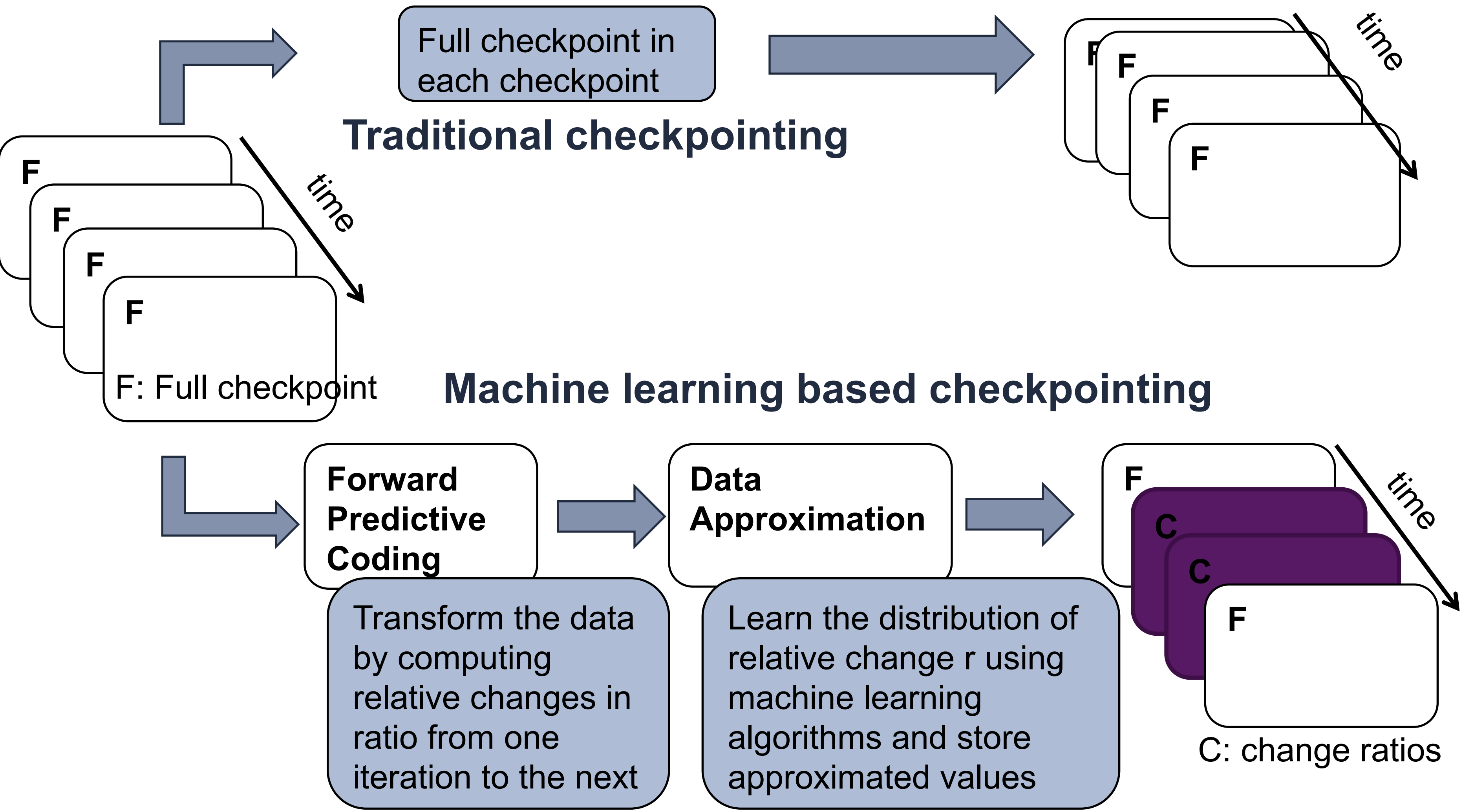



Relative Change between iteration 1 and 2 on climate CMIP5 rlus data

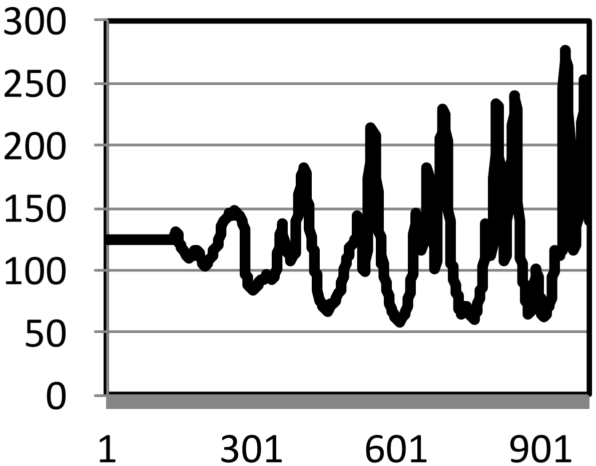
# Challenges

- How to learn patterns and distributions of relative change at scale?
- How to represent distributions at scale?
- How to bound errors?
- System Issues
  - data movement
  - I/O
  - Scalable software
  - Reconstruction when needed

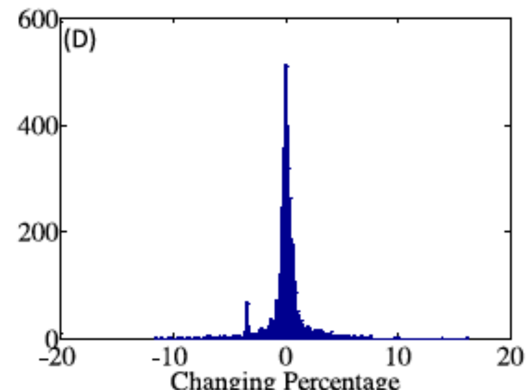
# NUMARCK Overview



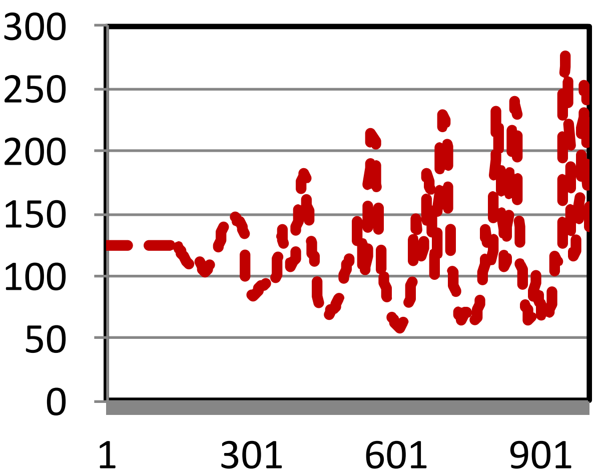
# NUMARCK: Overview




**Forward coding**




**~0.99 correlation!**
**0.001 RMSE**




Distribution Learning

# E.g., Distribution Learning Strategies

- Equal-width Bins (Linear)
- Log-scale Bins (Exponential)
- Machine Learning – Dynamic clustering

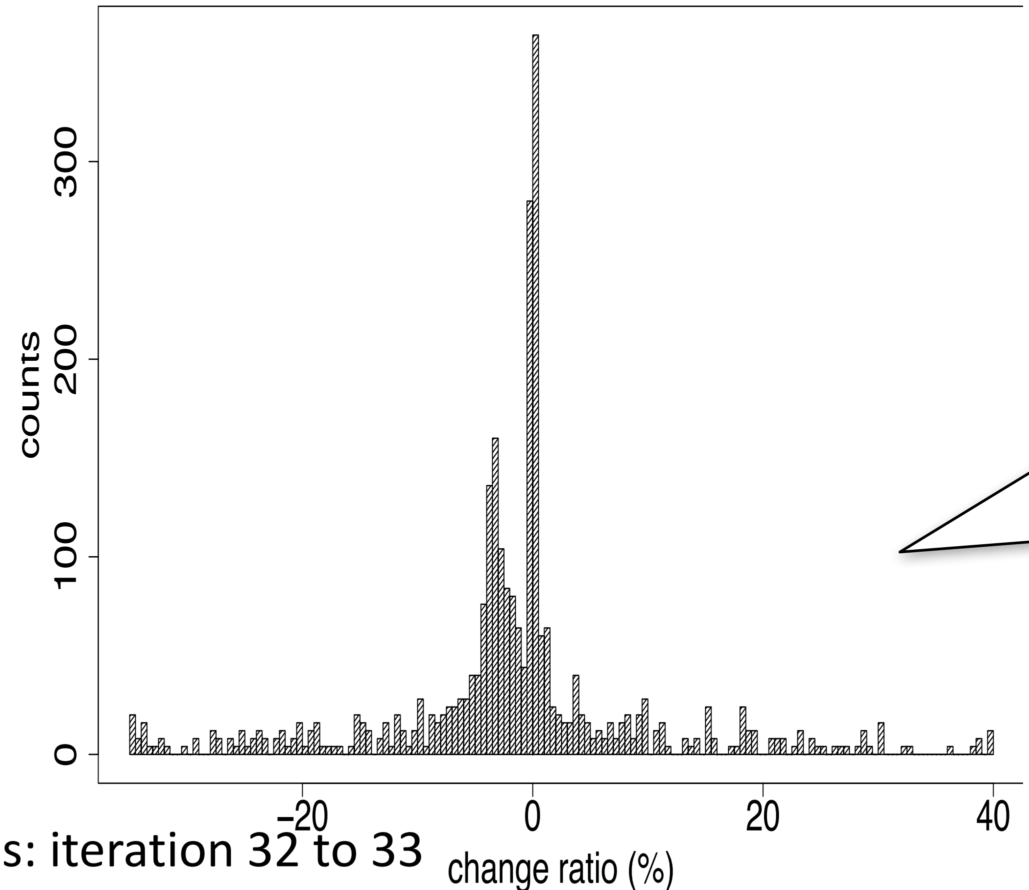
Number of bins or clusters depends on the bits designated for storing indices and error tolerance

examples

- index length (B): 8bits → **the number of clusters**
- tolerable error per point (E): 0.1% → **the width of each cluster**

# Equal-width distribution

In each iteration, partition value into 255 bins of **equal-width**. Each value is assigned to a corresponding bin ID ( represented by  the **center of bin**). If the difference between the original value and the approximated one is **larger than user-specified value** (0.1%), it is stored **as it is** (i.e., incompressible)



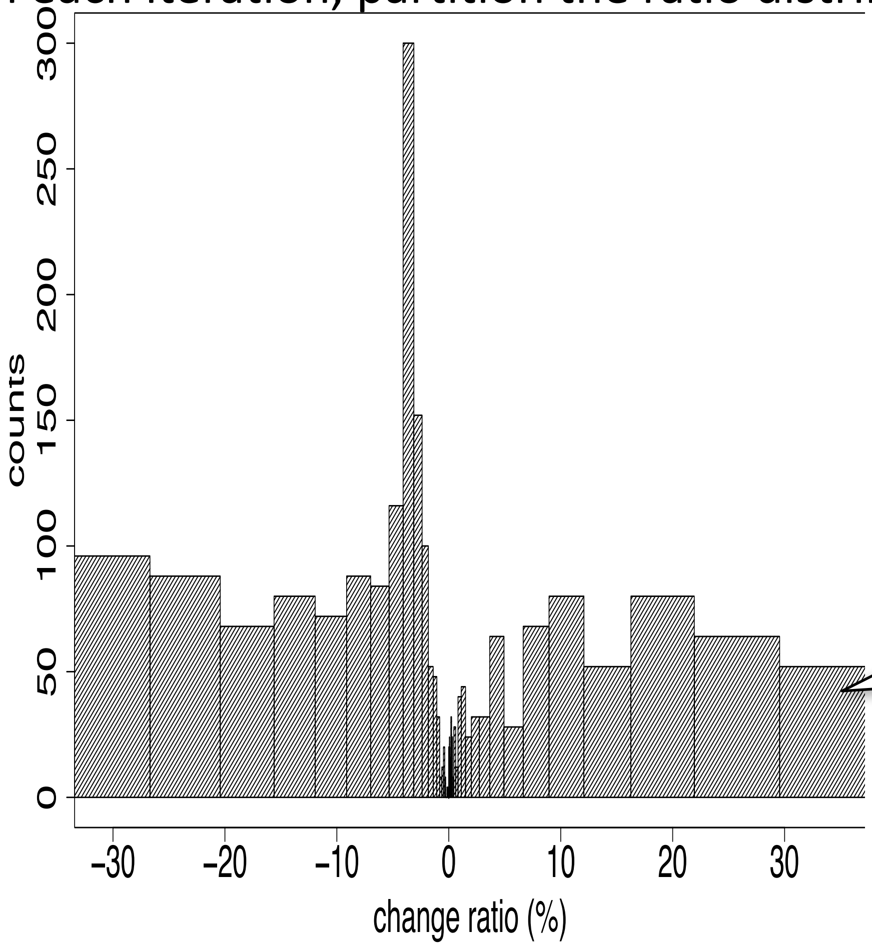

dens: iteration 32 to 33

**Pros**: Easy to Implement

**Cons**: (1) Can only cover range of 2*E*(2^B -1);
(2) Bin width: 2*E

# Log-scale Distribution

In each iteration, partition the ratio distribution into 255 bins of **log-scale width.**



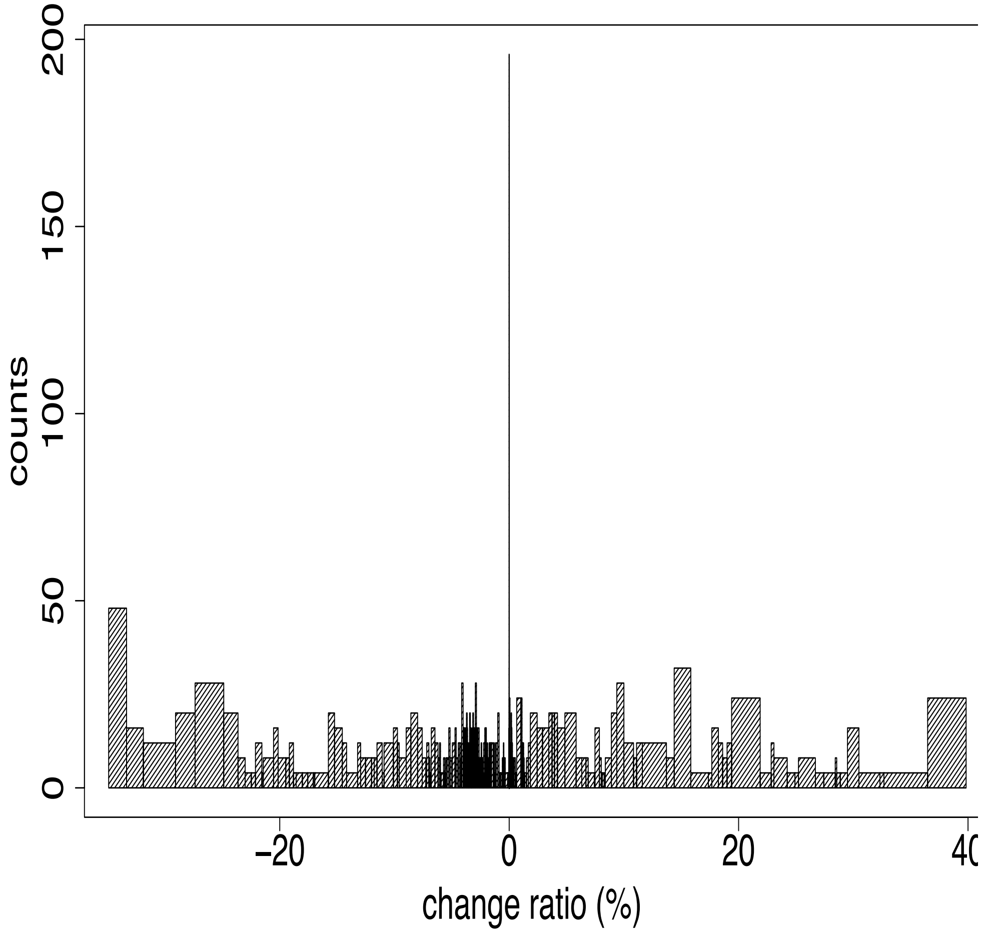

**Pros**: **cover larger ranger and more finer (narrower) bins**

**Cons**: **may not perform well for highly irregularly distributed data**

dens: iteration 32 to 33

# Machine Learning (Clustering-based) based Binning

In each iteration, partition the ratio data into 255 clusters using (e.g., K-means) clustering, followed by approximated values based on corresponding cluster's centroid value.




dens: iteration 32 to 33

# Methodology Summary

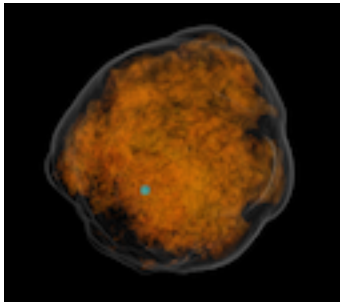
| | |
|---|---|
| Initialization | • this is the model, initial condition and metadata |
| Calculation | • Calculate the relative change |
| Learning Distributions | • Bin the relative change into N bins<br>• Index and Store bin IDs |
| Storage | •Store index, compress index<br>•Store exact values for change outside error bounds |
| Reconstruction | • Read last available complete checkpoint<br>• Reconstruct data values for each data point, can report the error bounds. |

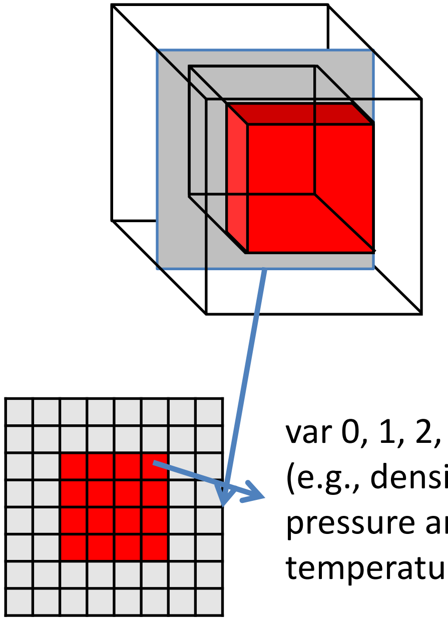
# NUMARCK Algorithm

- Change ratio calculation
  - Calculate element-wise change ratios
- Bin histogram construction
  - Assign change ratios within an error bound into bins
- Indexing
  - Each data element is indexed by its bin ID
- Select top-K bins with most elements
  - Data in top-K bins are represented by their bin IDs
  - Data out of top-K bins are stored as is
- (optional) Apply lossless GNU ZLIB compression on the index table
  - Further reduce data size
- (optional) File I/O
  - Data is saved in self-describing netCDF/HDF5 file

# Experimental Results: Datasets

- FLASH code is a modular, parallel multi-physics simulation code: developed at the FLASH center of University of Chicago
  - It is a parallel adaptive-mesh refinement (AMR) code with block-oriented structure
  - A block is the unit of computation
  - The grid is composed of blocks
  - Blocks consists of cells: guard and interior cells
  - Cells contains variable values
- CMIP - supported by World Climate Research Program: (1) Decadal Hindcasts and predictions simulations; (2) Long-term simulations; (3) atmosphere-only simulations.



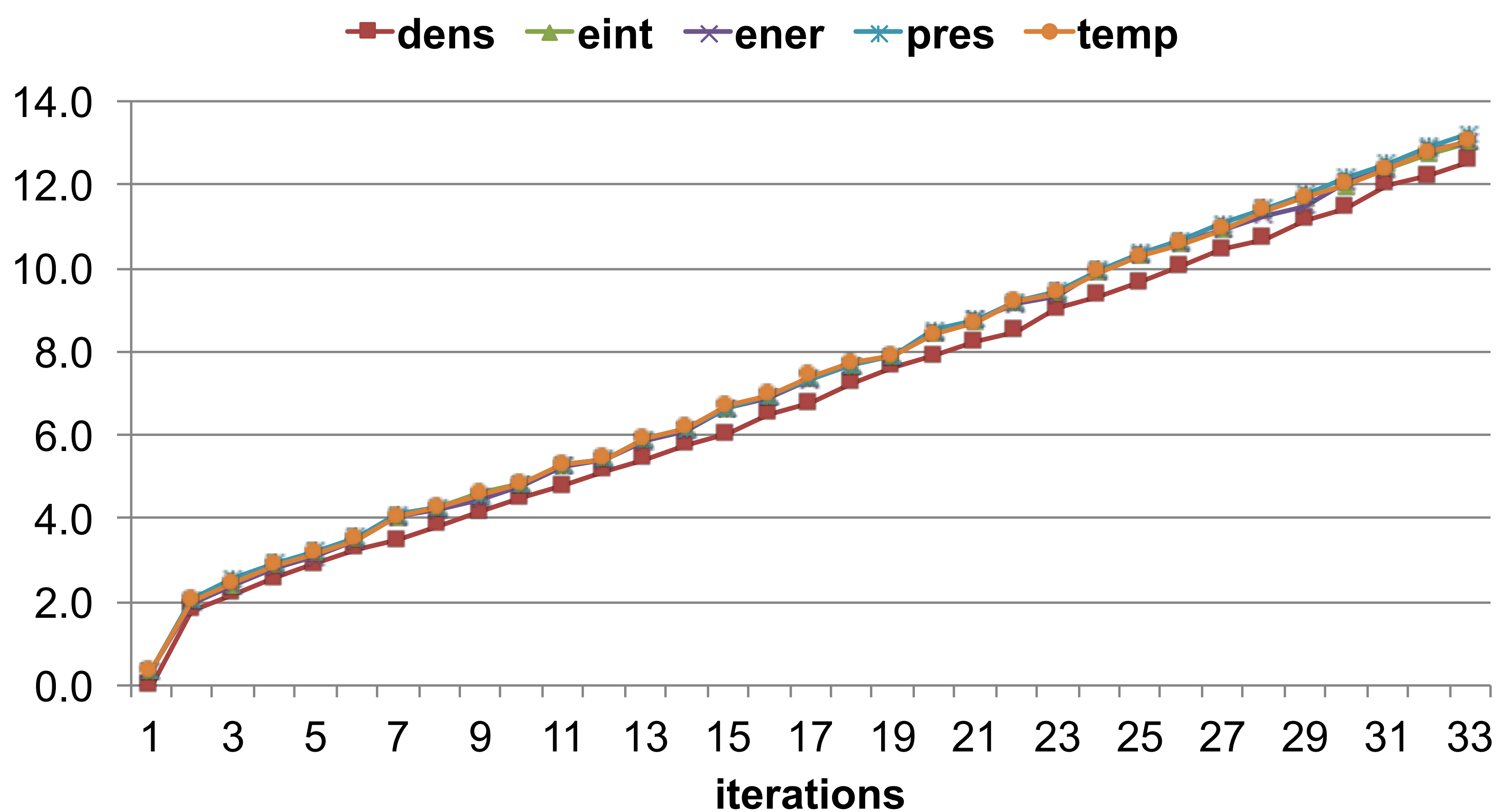


# Evaluation metrics

- Incompressible ratio
  - % of data that need to be stored as exact values because it would be out of error bound if approximated
- Mean error rate
  - Average difference between the approximated change ratio and the real change ratio for all data
- Compression ratio
  - Assuming data D of size |D| is reduced to size |D'|, it is defined as:

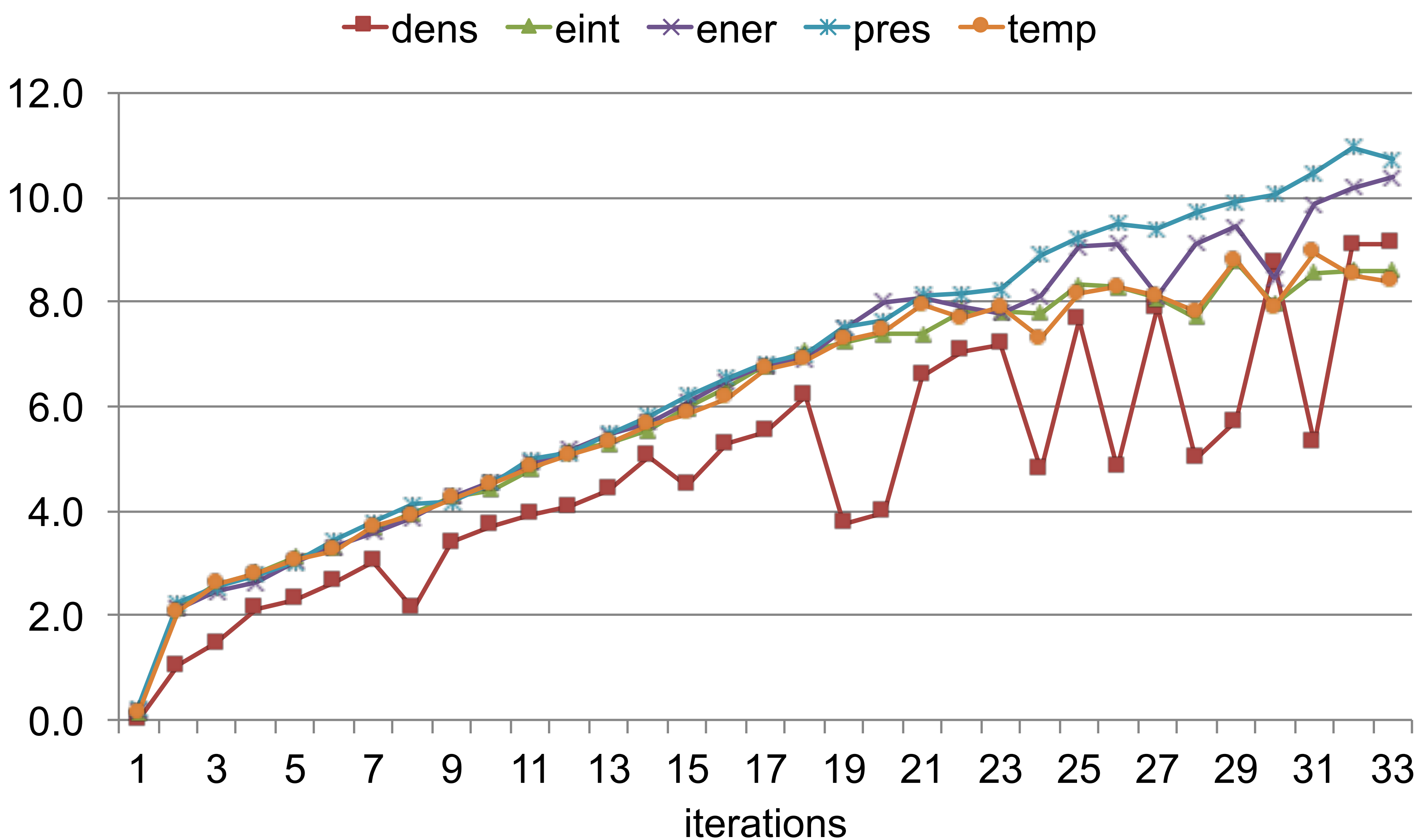
$$\frac{|D| - |D'|}{|D|} \times 100$$

# Incompressible Ratio: Equal-width Binning



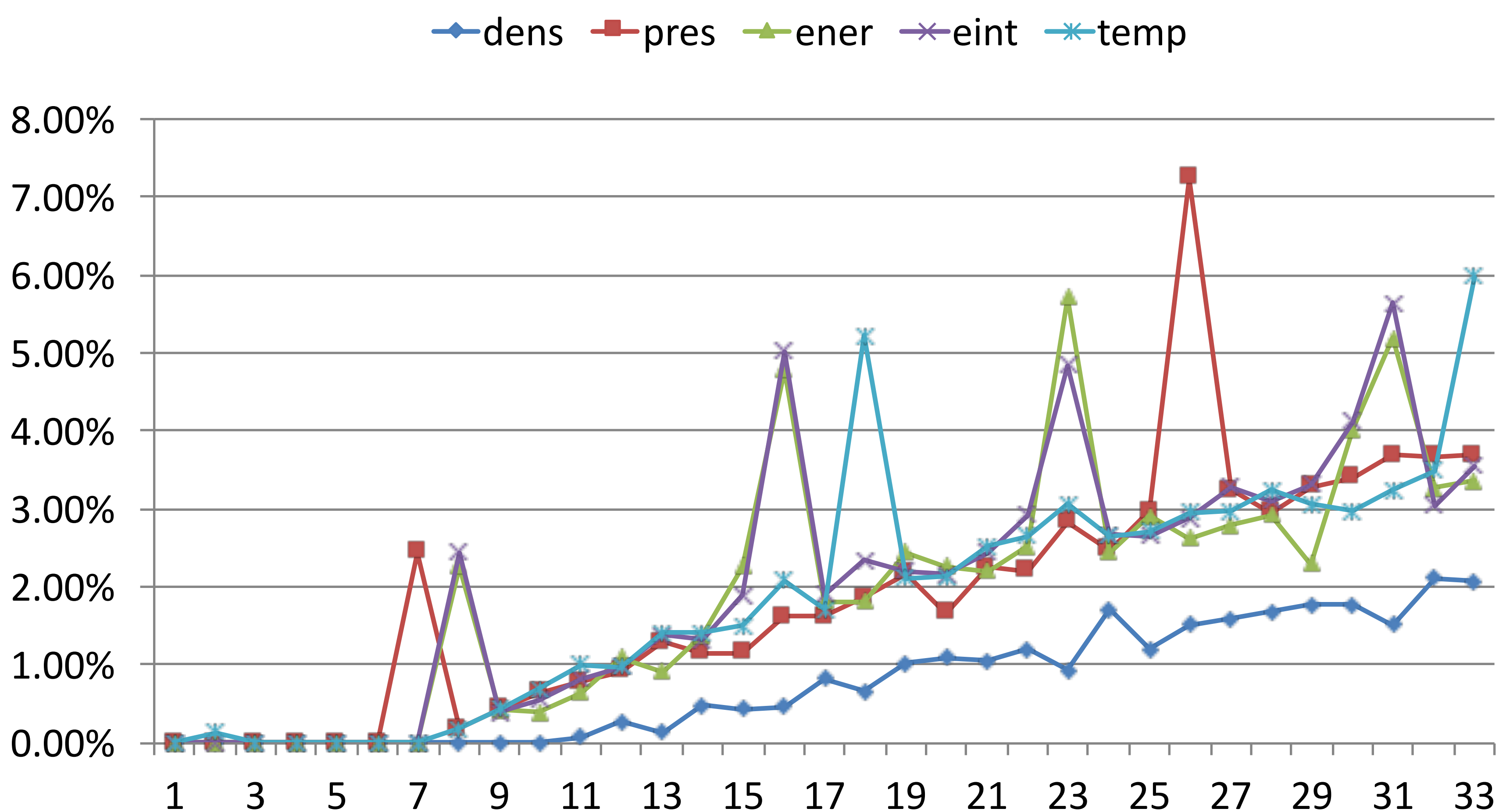

**FLASH dataset, 0.1% error rate**

# Incompressible Ratio: Log-scale Binning



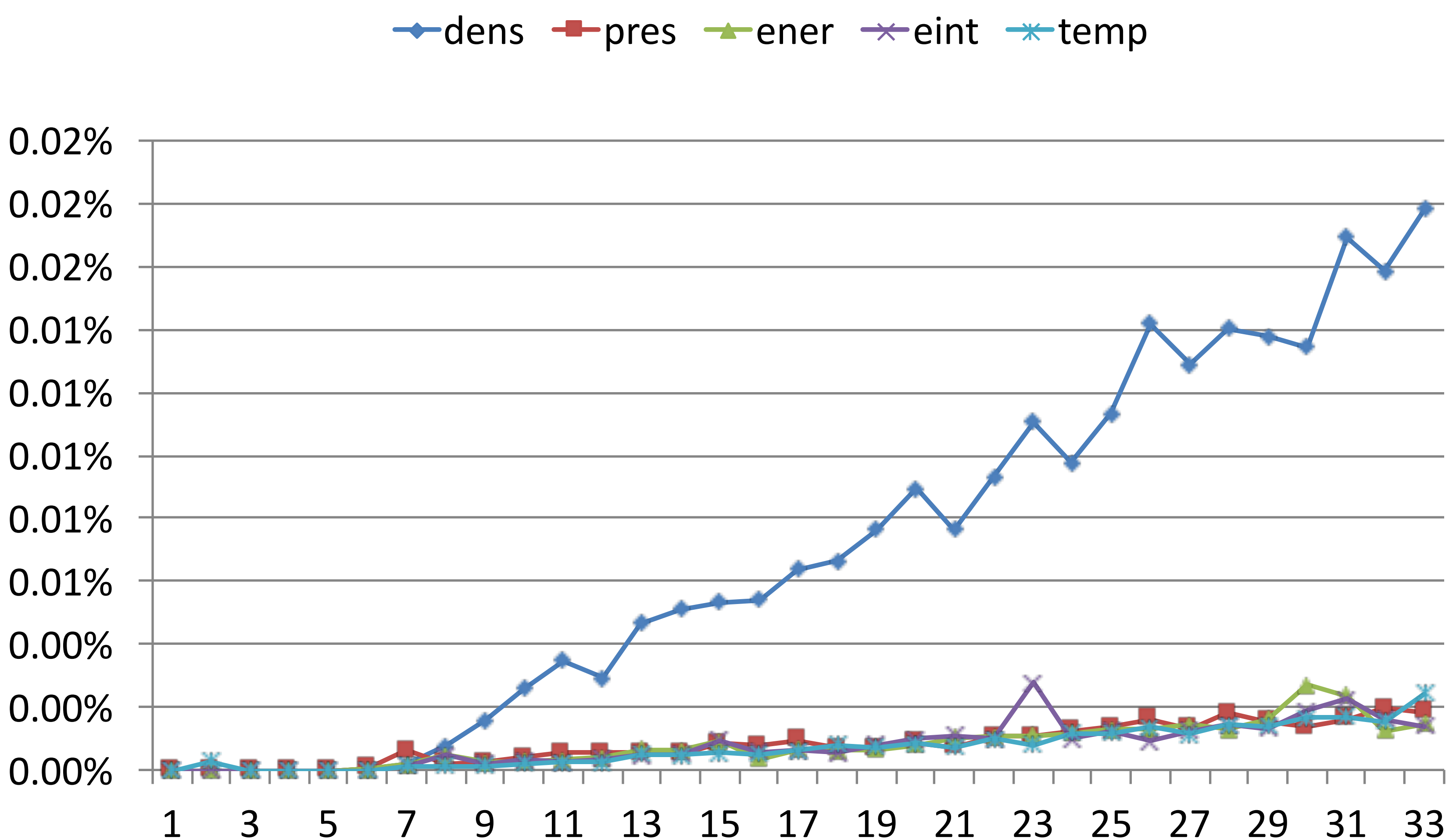

FLASH dataset, 0.1% error rate

# Incompressible Ratio: Clustering-based Binning



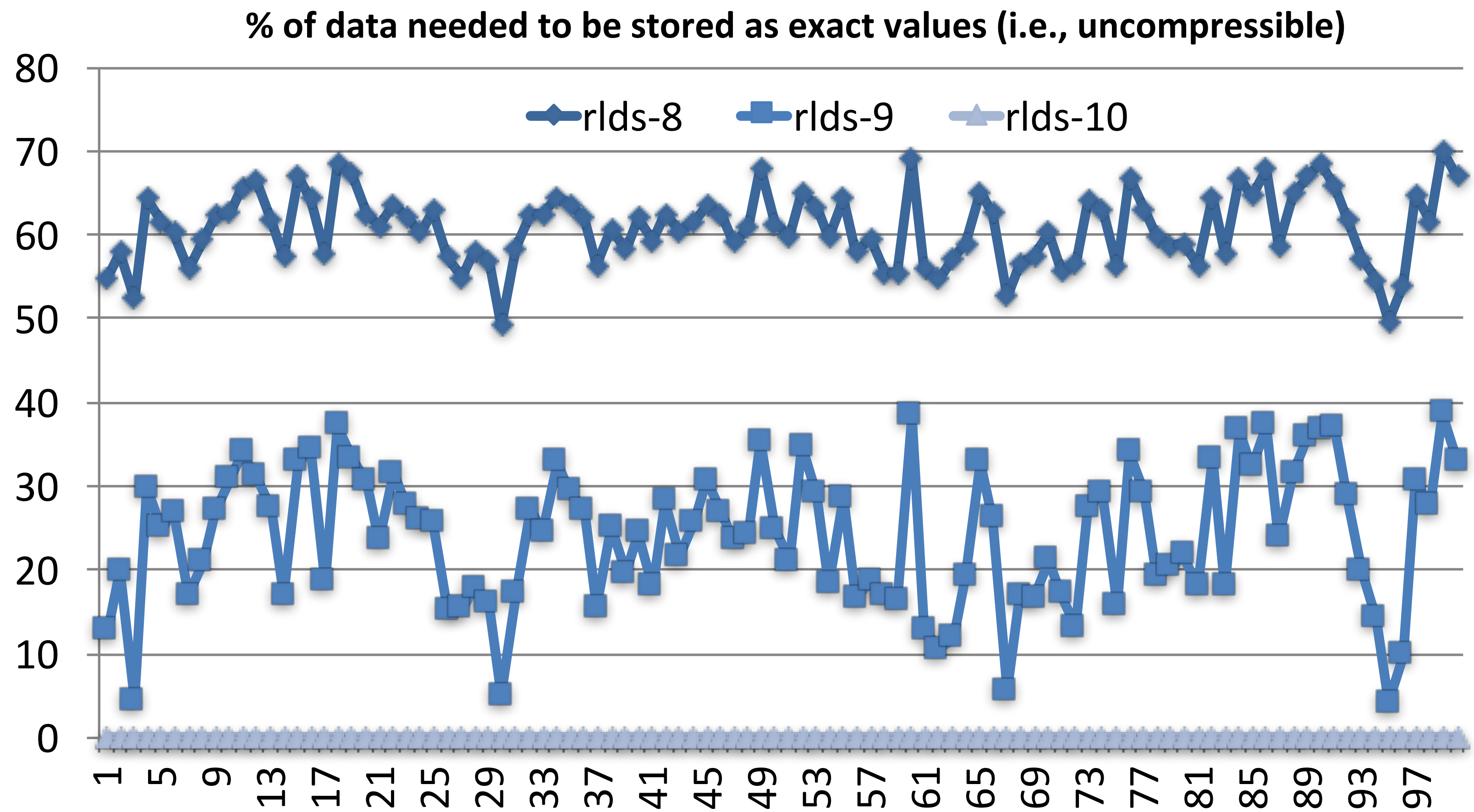

FLASH dataset, 0.1% error rate

# Mean Error Rate: Clustering-based



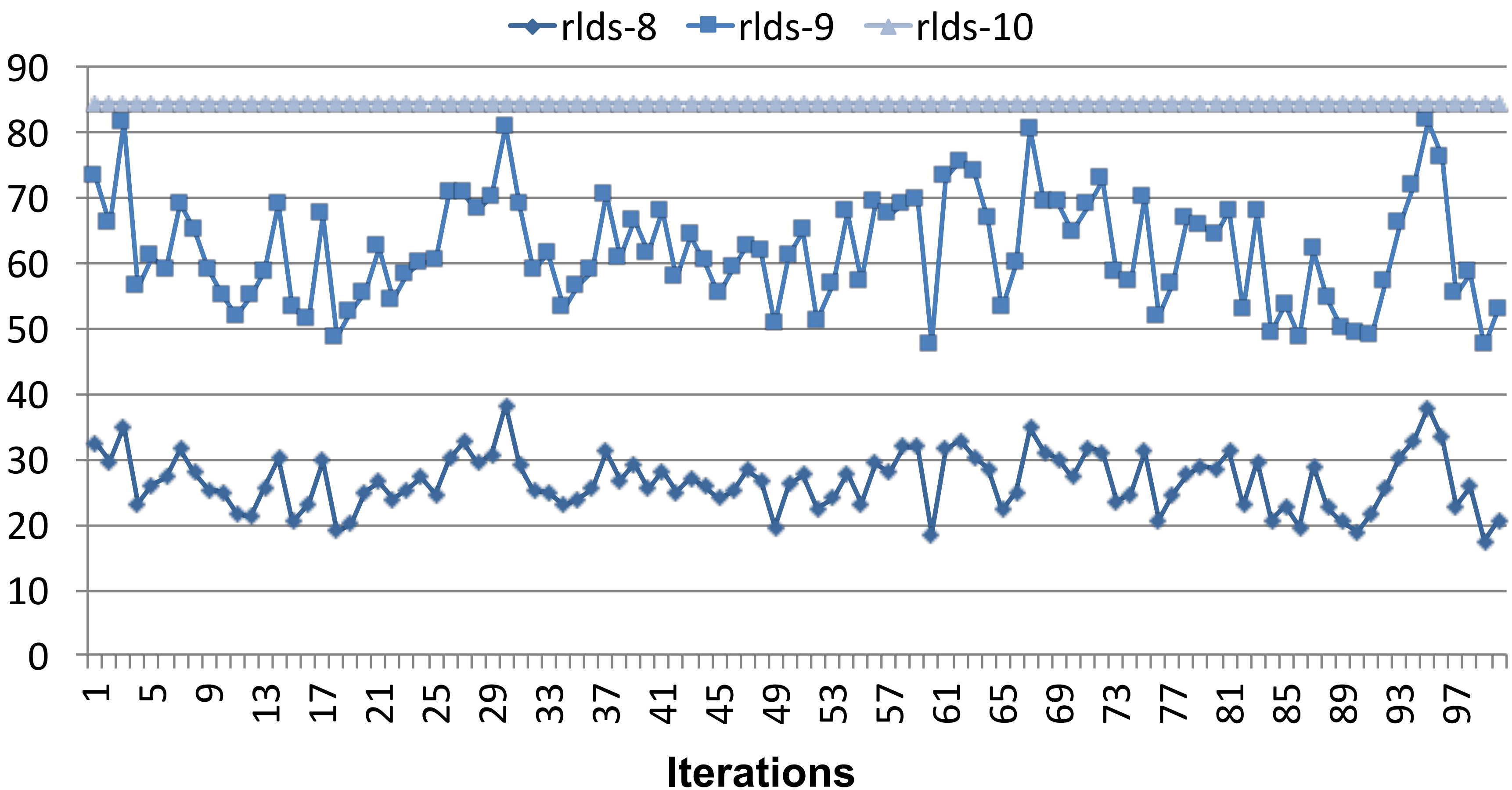

FLASH dataset, 0.1% error rate

# Increasing Index Size: **Incompressible Ratio**



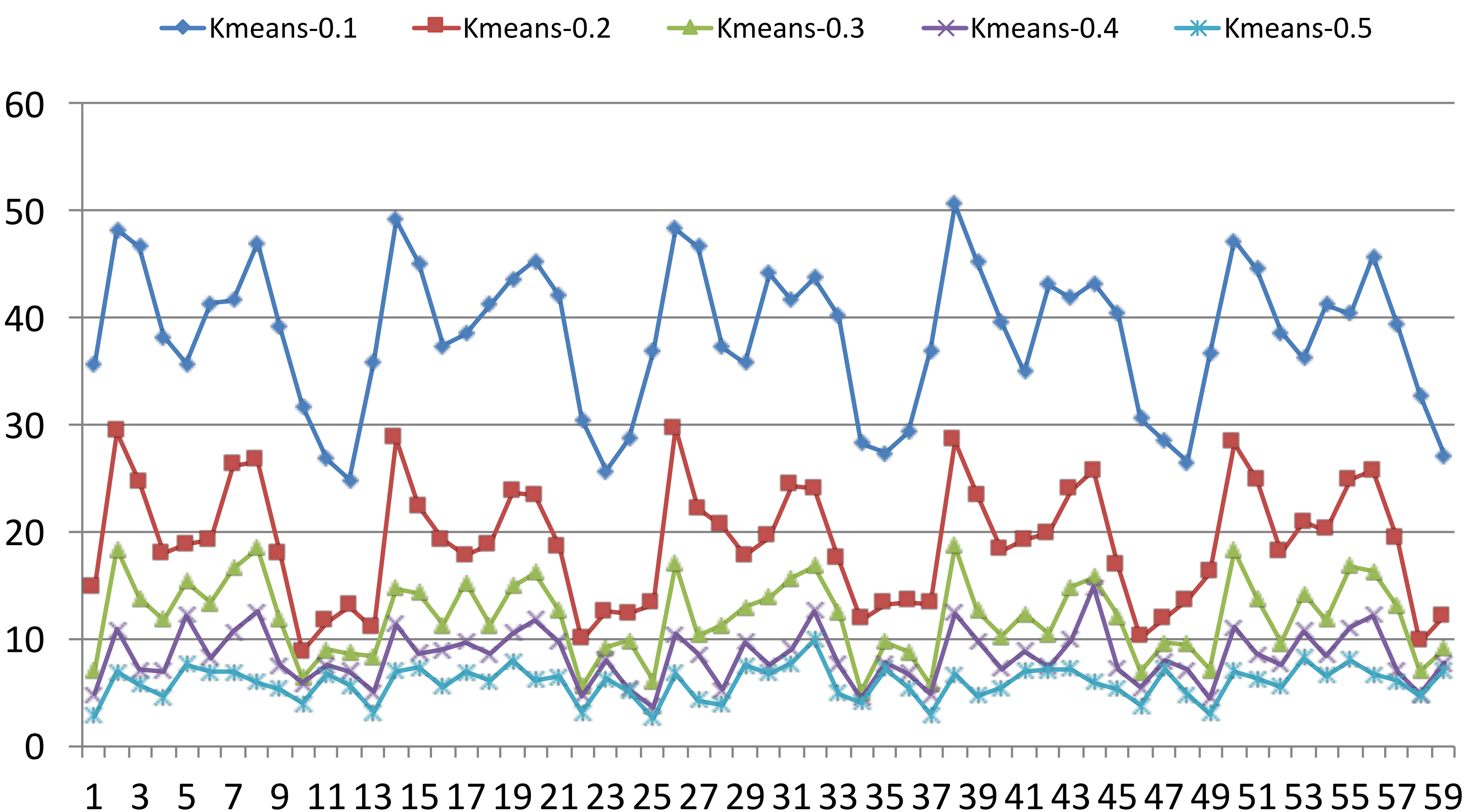

Increasing bin sizes (8-bit to 10-bit) reduces % of incompressible significantly.
Note: rlds is the most difficult to compress with 8-bit

# Different Approximations: **Compression Ratio**




Increasing bin sizes (8-bit to 10-bit) increases compression ratio significantly.

# Different Tolerable Error Rates: Incompressible Ratio (0.1% - 0.5%)



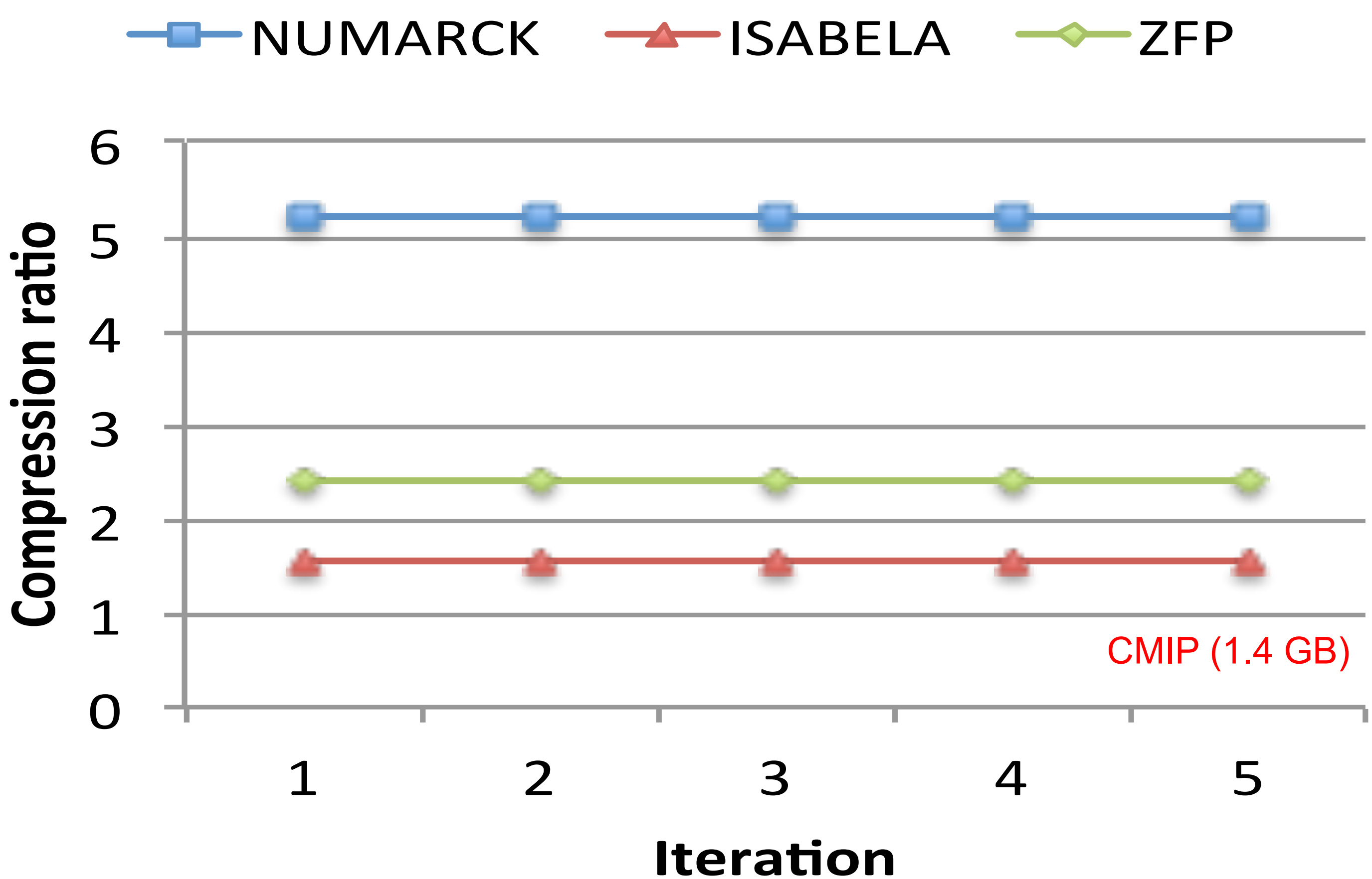
# Scaling - Experimental Settings

| Name of data set | Application | Domain | Size per iteration | Variable dimension | Variable size |
|---|---|---|---|---|---|
| Sedov | FLASH | Astrophysics | 15MB | 165*32*32*1 | 1.3MB |
| Stir-1 | FLASH | Astrophysics | 3.7GB | 64*157*157*157 | 945MB |
| Stir-2 | FLASH | Astrophysics | 296GB | 1024*314*314*157 | 59GB |
| Stir-3 | FLASH | Astrophysics | 2.4TB | 8192*314*314*157 | 472GB |
| ASR | ASR | Climate | 103MB | 29*320*320 | 11MB |
| CMIP | CMIP3 | Climate | 19GB | 42*2400*3600 | 1.4GB |

- Data sets and environment:
  - FLASH datasets
    - SuperMUC at Leibniz Supercomputing Centre, Germany, a parallel computer consists of 9216 nodes (16 cores per node)
    - We used up to 12,800 cores in our experiments
  - Others
    - A Linux machine, 2 quad-core CPUs (32 GB memory)

# Compression ratios

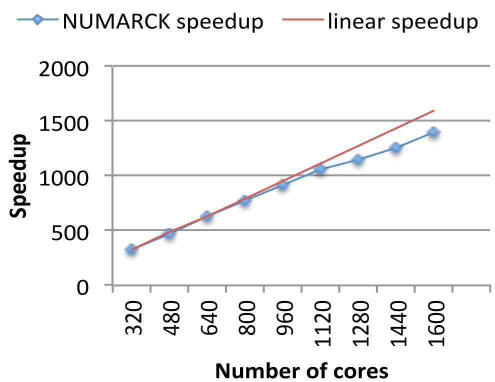
- Compared with lossy compression algorithms: ZFP (LLNL), ISABELLA (NCSU)



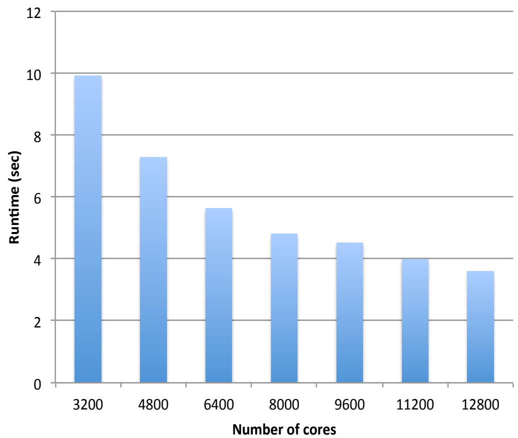
# Scalability Experiments

FLASH datasets (turbulence stirring test)

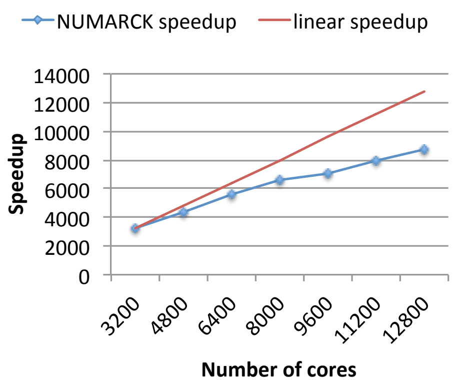
- Stir-2 (59GB) data
  - Numbers of cores: 1600
  - Speed-up: 1404
  - Time: 2.655 sec
  - Original I/O time: 13.2 sec/iteration
- Stir-3 (472GB) dataset
  - Number of cores:12800
  - Speed-up: 8788
  - Time: 3.610 sec
  - Original I/O time: 18.0 sec



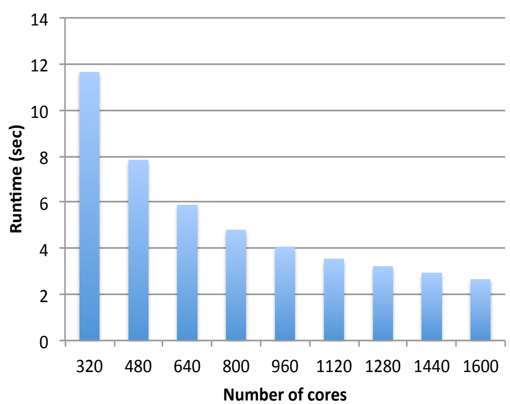

Speedup of Stir-2




Runtime of Stir-2




Speedup of Stir-3




Runtime of Stir-3

# Open Problems and Challenges

- Optimize and/or create new machine learning algorithms
  - for higher compressions and more accurate representation
  - Scalable implementation
  - Learning from historical results to optimize the "learning step"

  for minimizing data movement and power
  - Adaptation for anomaly detection (for resilience and analysis)
- Use of memory hierarchy and local SSDs
- Incorporation into pNetCDF etc and libraries
- I/O optimizations

# THANK YOU!

**Alok Choudhary**
Henry and Isabel Dever Professor
EECS and Kellogg School of Management
Northwestern University
choudhar@eecs.northwestern.edu