IPComp: Interpolation Based Progressive Lossy Compression for Scientific Applications

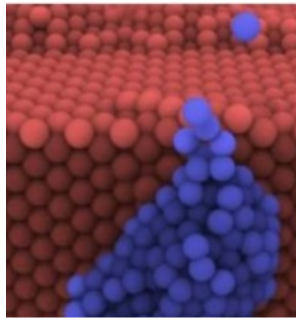
Zhuoxun Yang (***Florida State University)***

Sheng Di, Longtao Zhang, Ruoyu Li, Ximiao Li, Jiajun Huang, Jinyang Liu, Franck Cappello, Kai Zhao

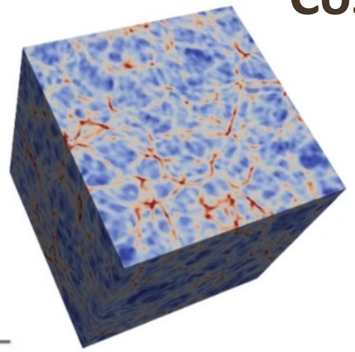
Lossy compression for Scientific Datasets

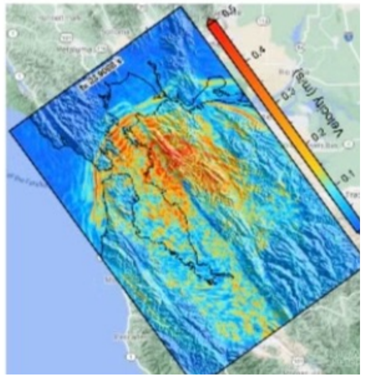
- Scientific Datasets:
  floating-point or integer values
- significantly reducing storage
- avoiding recomputation cost
- accelerating checkpoint/restart
- accelerating the I/O performance

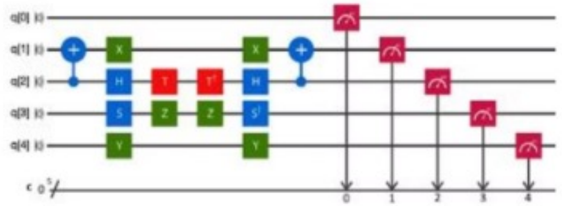
Molecular Dynamics
(LAMMPS, GROMACS)



Cosmology(NYX)







Quantum Simulation
(Q-Tensor)

Seismology(SCOPED)

Error-bounded Lossy compression

Progressive Lossy Compression

Lossy compression for Scientific Datasets

Error-bounded Lossy compression

- allows to control the data distortion.
  common **distortion metrics** includes:
  Absolute Error, Relative Error, Peak Signal-to-Noise Ratio (PSNR), Root Mean Square Error (RMSE) / Mean Absolute Error (MAE)
- classic general-purpose error-bounded lossy compressors:
  SZ, ZFP, etc.
- emerging tailored lossy compressors:
  SPERR, AESZ, FAZ, MDZ, MGARD, etc.

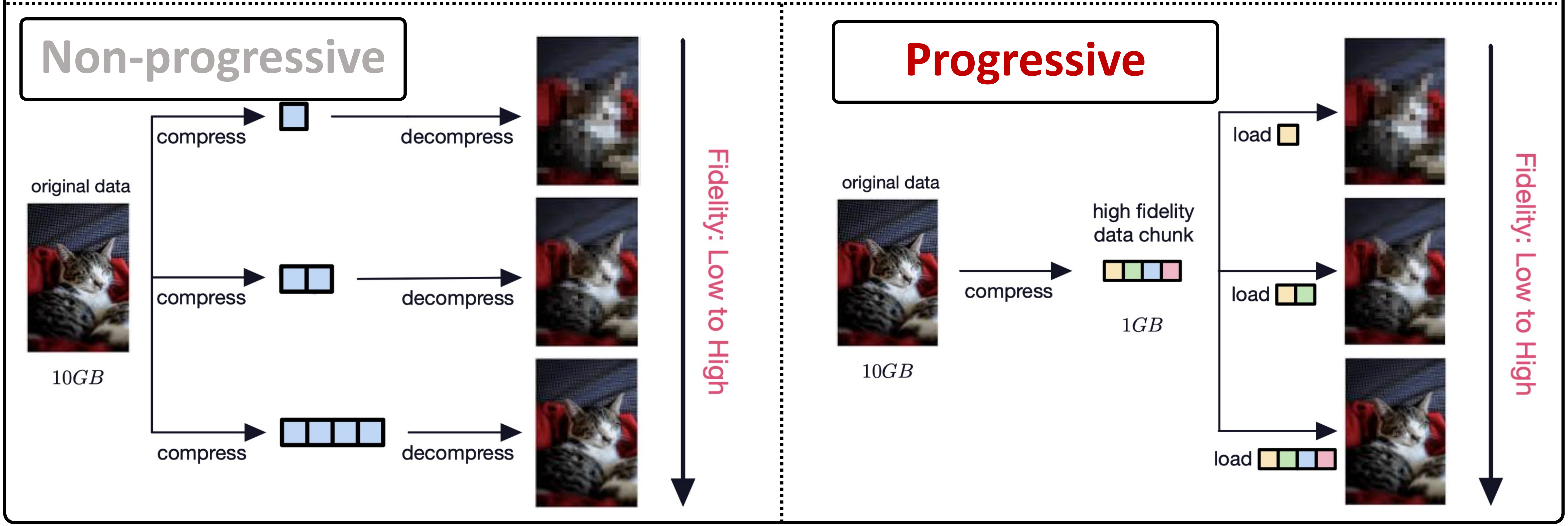
Progressive Lossy Compression

Lossy compression for Scientific Datasets

Error-bounded Lossy compression

Progressive Lossy Compression

➢ Multi-Fidelity Delivery

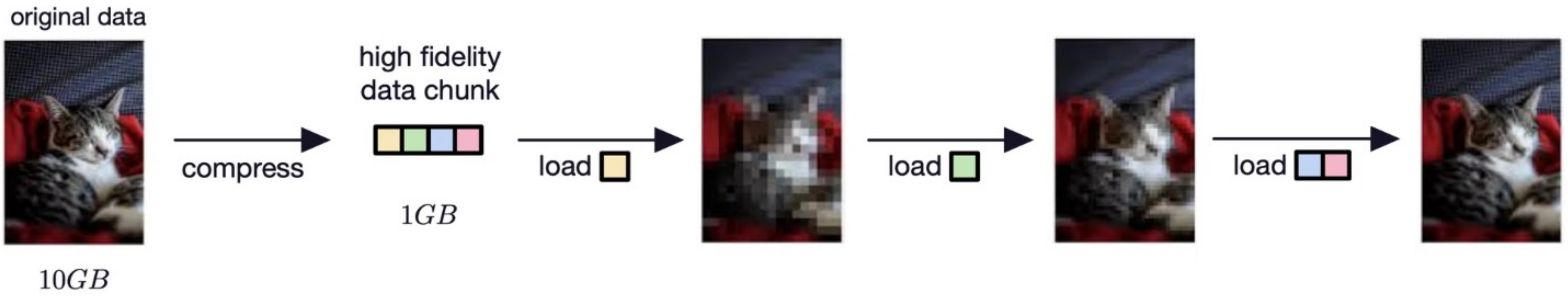


Lossy compression for Scientific Datasets

Error-bounded Lossy compression

Progressive Lossy Compression

- Multi-Fidelity Delivery
- Progressive Reconstruction



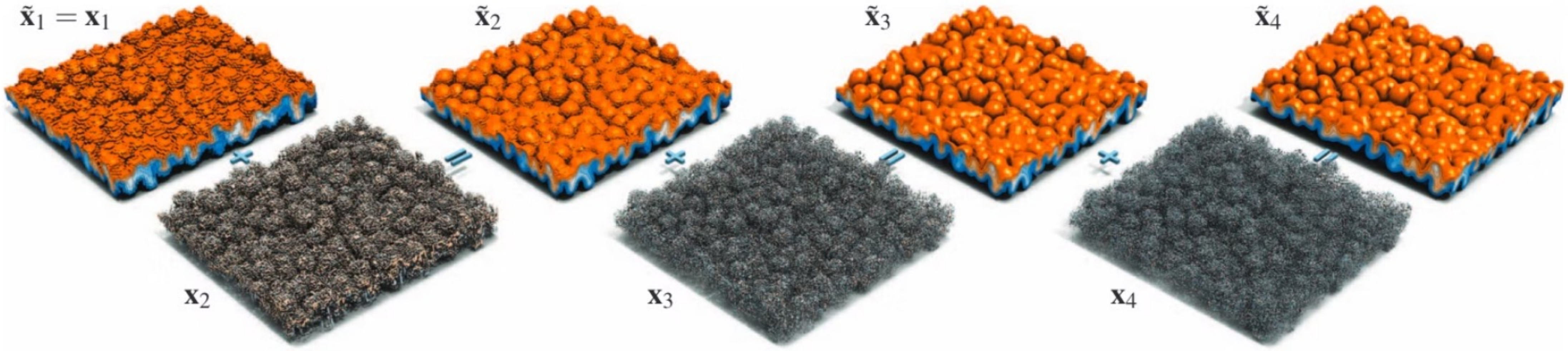
Limitations of Current Approaches

Two types of Approaches:

- **The first type: Embedded Progressive Compressors**
  - modifies the compression and decompression workflows of existing general-purpose lossy compressors to support progressive features.
  - suffer from **compression ratio trade-offs** and **degraded performance**, due to the added complexity. A representative example is PMGard, the progressive variant of MGARD.

- **The second type: Residual Compressors**
  - general framework for progressive compression
  - **loss of compression efficiency**
  - Significant **operational overhead** due to repeated compression and decompression of residuals
  - **Does not support arbitrary error bounds** — users can only decompress data at a few pre-specified precision levels.

- **Low Compressibility**
- **Low Performance**
- **High Overhead**
- **Limited Flexibility**



Our Contribution: IPComp

High Compression Ratio

- Up to **487%** higher compression ratio
- interpolation-based compression approach
- resolved the issue of inter-level dependencies from interpolation
- a lightweight and high-performance encoding scheme
- better compressibility under progressive compression scenarios, where traditional Huffman encoding becomes less effective
- **Lower reconstruction errors** compared to **Lorenzo** prediction

Low Operational Cost

- only **one-time** compression and decompression, in contrast to the residual approach which involves multiple iterations.
- The user simply provides the target error bound or bitrate as input.

Fast Compression/Decompression Speeds

- high throughput in both compression and decompression

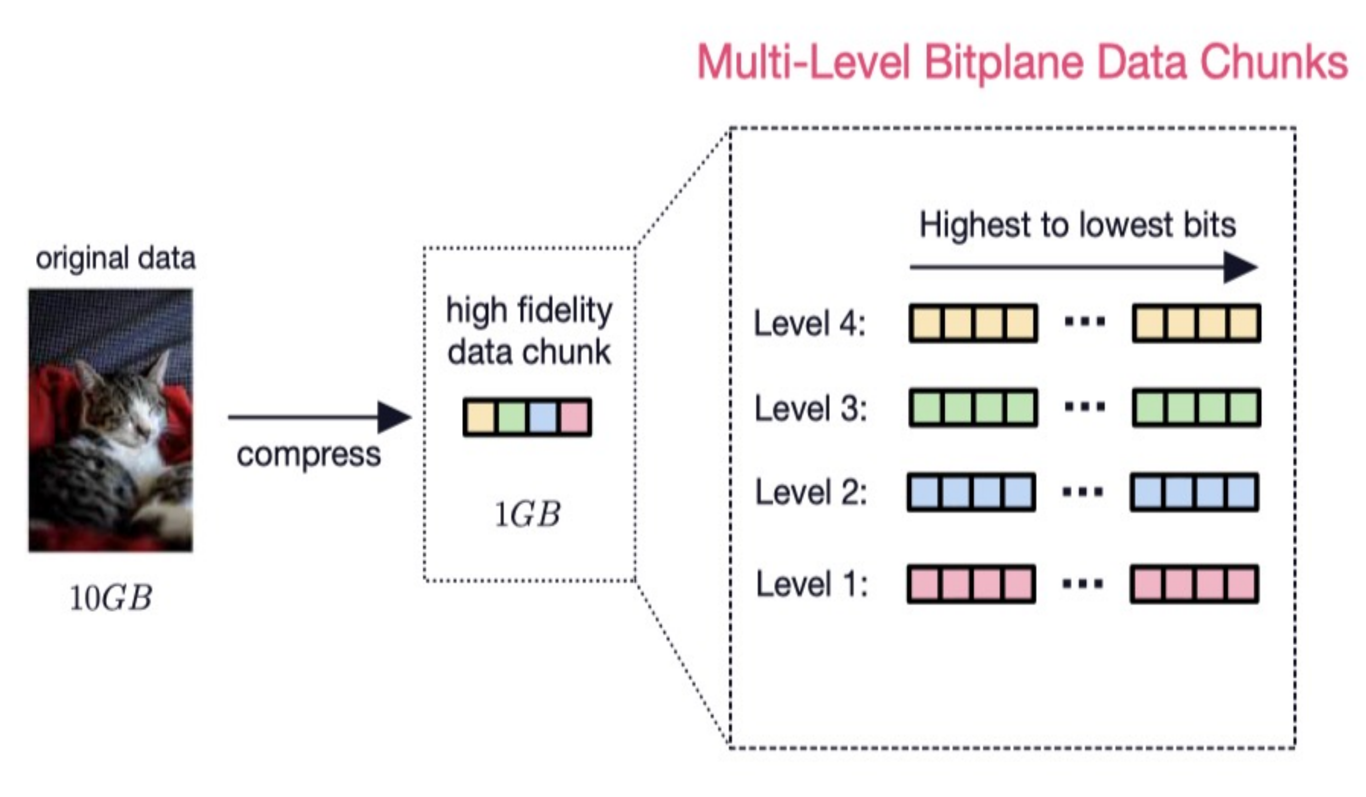
Arbitrary Reconstruction Accuracy

- splits the data into small chunks by **combining level-wise and bitplane-wise partitioning**
- we develop a **rigorous error prediction model** that can accurately estimate the error introduced when only a subset of the data is reconstructed.

Design of IPComp

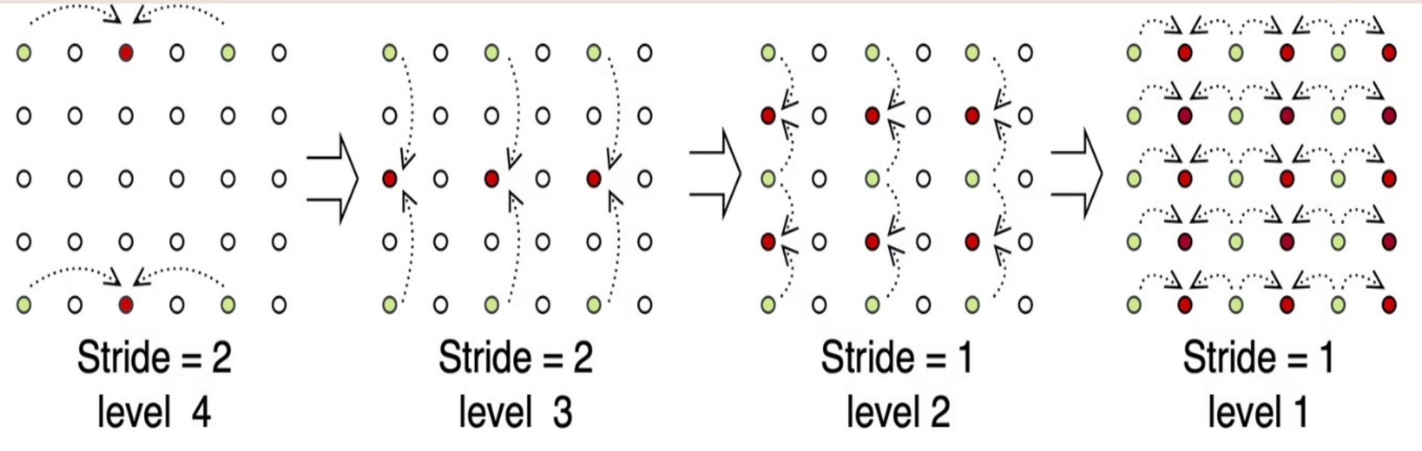
Overview

- Compression(Refactorization): **Multi-Level Bitplane Data Chunk**



- Decompression(Reconstruction)

Interpolation-Based Algorithm



- Each level corresponds to a set of data points, which are the newly added points following a halving stride pattern, as illustrated in the figure
- There are dependencies between levels: data in a finer level is predicted using data from a coarser (higher) level, using either**linear** or **cubic interpolation**.
- Adopted by SZ3. High compression ratios
- Lower reconstruction errors compared to **Lorenzo** prediction

Design of IPComp

Overview

- Compression(Refactorization)
- Decompression(Reconstruction):

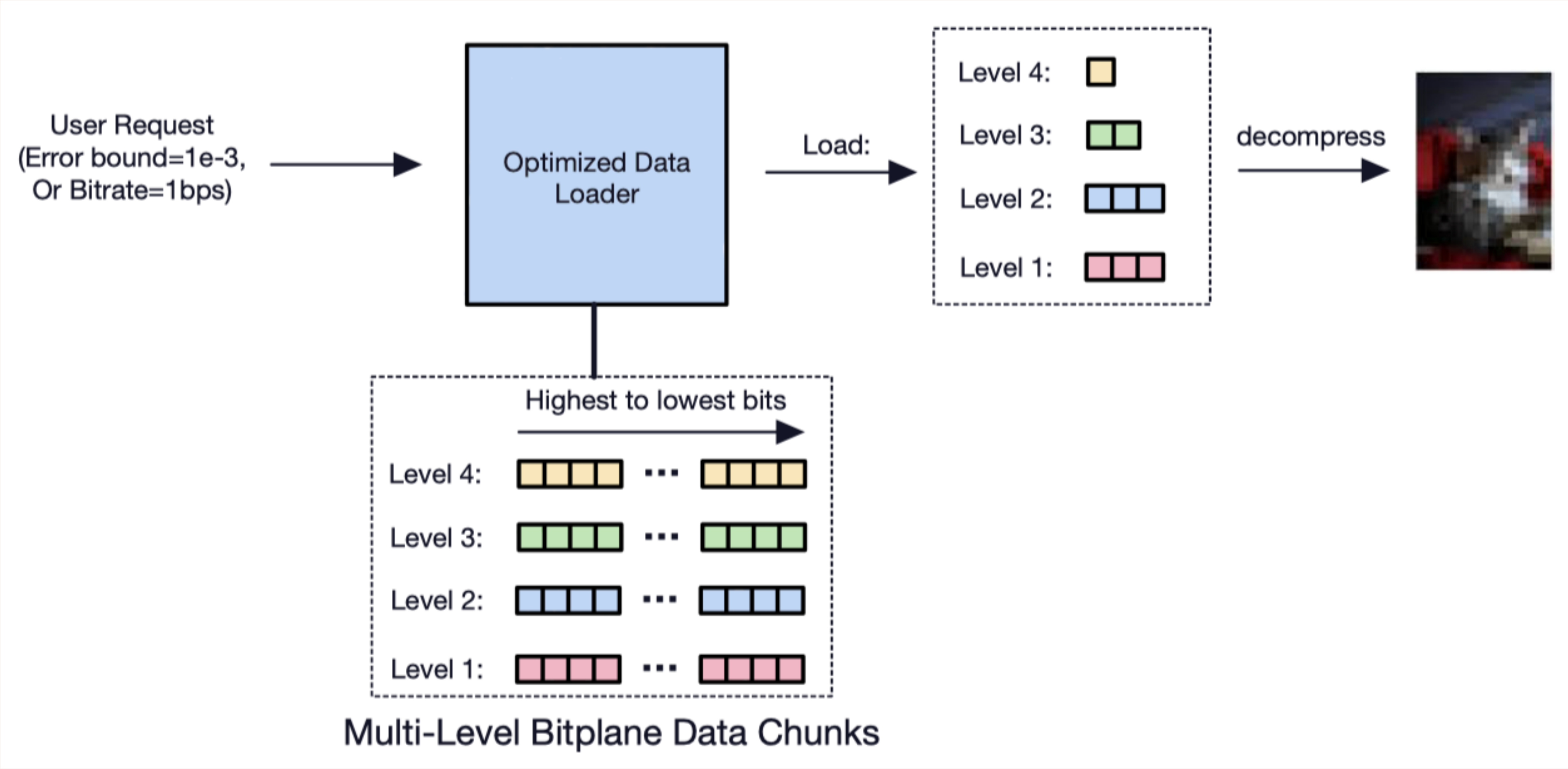
Arbitrary Fidelity Reconstruction

Optimized Data Loader

Algorithm 1 Reconstruction Algorithm

Require: $bitplaneList[L]$

$\hat{x} \leftarrow 0$
$\Pi_L \hat{x} \leftarrow P_L(0)$
for $l \leftarrow L-1$ **downto** $L_p + 1$ **do**
  $q_l \leftarrow$ **decode**$(bitplaneList[l])$
  $\hat{y}_l \leftarrow$ **dequantizition**(q_l)
  $\hat{x}_l \leftarrow$ **Predict**$(\hat{x}, \hat{y}_l)$
end for
$\Delta \leftarrow 0$
for $l \leftarrow L_p$ **downto** 1 **do**
  $q_l \leftarrow$ **decode**$(bitplaneList[l])$
  $\hat{y}_l \leftarrow$ **dequantizition**(q_l)
  $\hat{x}_l \leftarrow$ **Predict**$(\Delta, \hat{y}_l)$
  $\Delta_l \leftarrow$ **Predict**$(\Delta, \hat{y}_l)$
end for
return $\hat{x}$



Design of IPComp

Overview

- ➢ Compression(Refactorization)
- ➢ Decompression(Reconstruction):

Incremental Data Loading

Optimized Data Loader

Algorithm 2 Incremental Reconstruction Algorithm

Require: $\hat{x}^{(\mathrm{old})}$, $bitplaneList[L]$

$\hat{x}^{(\mathrm{new})} \leftarrow \hat{x}^{(\mathrm{old})}$

$\Delta \leftarrow 0$

for $l = L_p$ **to** 1 **do**

  $q_l \leftarrow$ **decode**$(bitplaneList[l])$

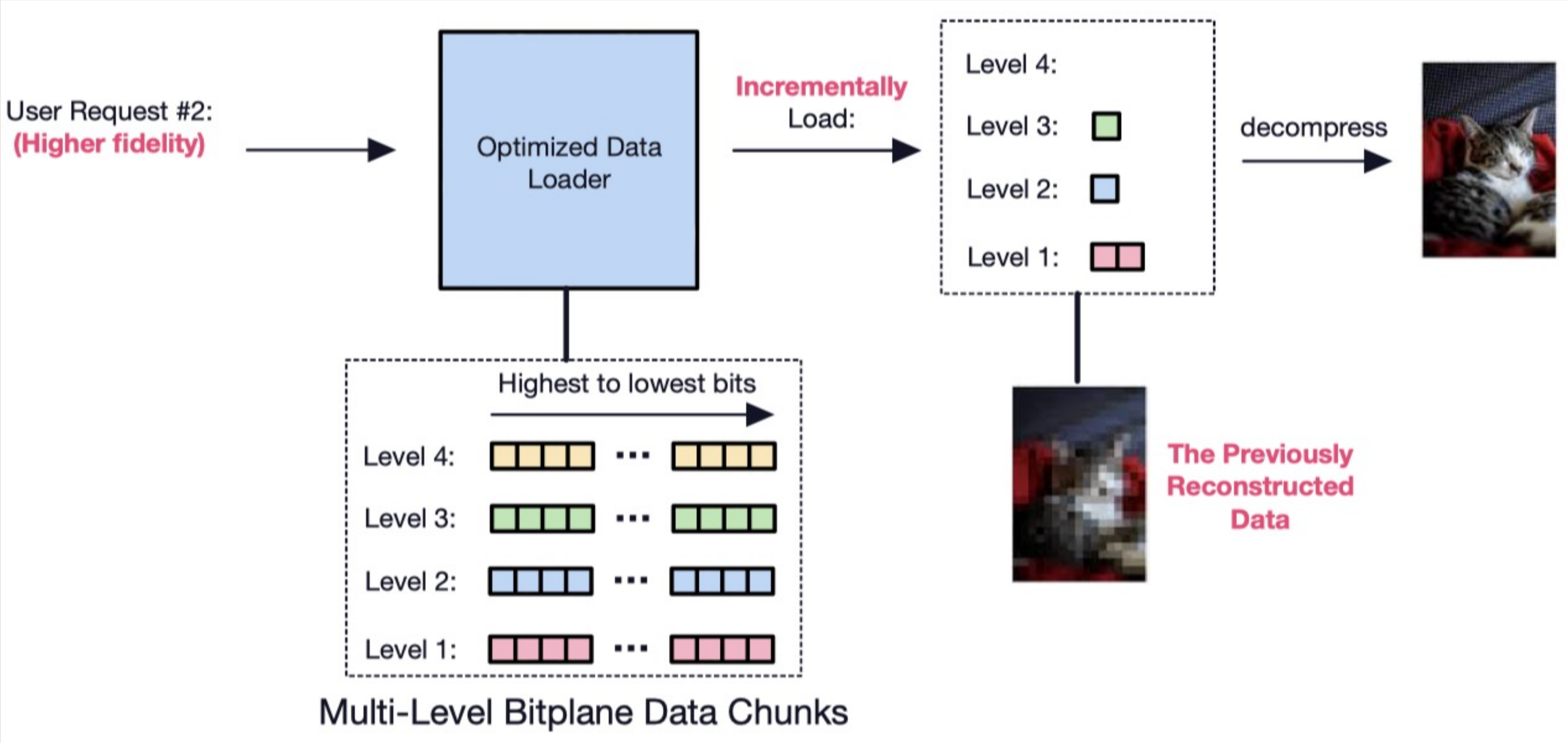
  $\hat{y}_l =$ **dequantization**(q_l)

  $x_l^{(\mathrm{new})} \leftarrow$ **Predict**$(\Delta, \hat{y}_l)$

  $\Delta_l \leftarrow$ **Predict**$(\Delta, \hat{y}_l)$

end for

return $\hat{x}^{(\mathrm{new})}$

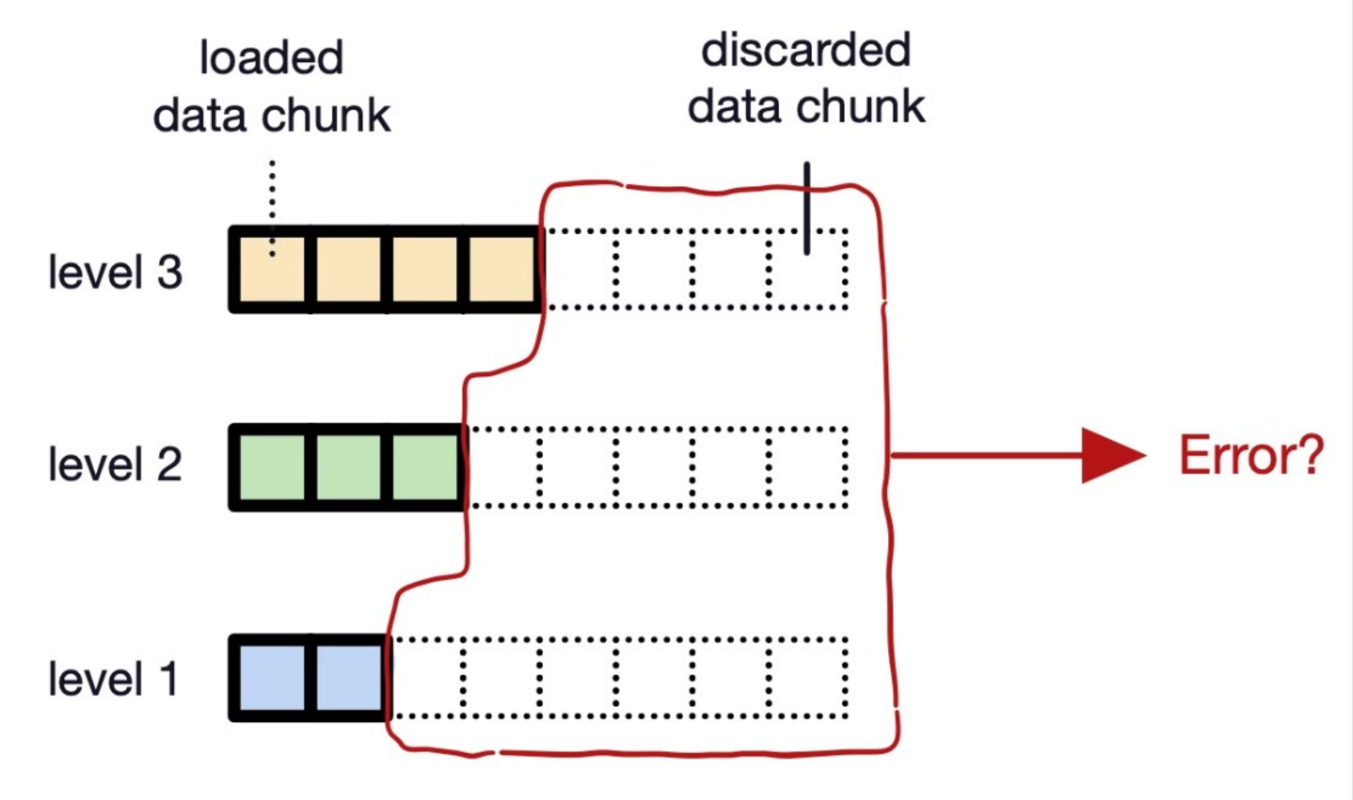


Design of IPComp

Overview

- ➢ **Compression(Refactorization)**
- ➢ **Decompression(Reconstruction):**

Optimized Data Loader: Error Formulation



- ➢ Recall: Data in a finer level is predicted using data from a coarser (higher) level
- ➢ The dependencies across levels can result in the **accumulation of errors**.
- ➢ **Theorem: Error Upper Bound**

THEOREM 1. *The L_∞ error in progressive retrieval can be bounded based on the information loss due to the unloaded bitplanes.*

$$\|x - \hat{x}\|_\infty \leq \sum_{l=0}^{L-1} p^l \|\delta y_{l+1}\|_\infty + eb \quad (5)$$

- ➢ The problem becomes **how to load data chunks** under a given error bound such that the **amount of data loaded is minimized**.

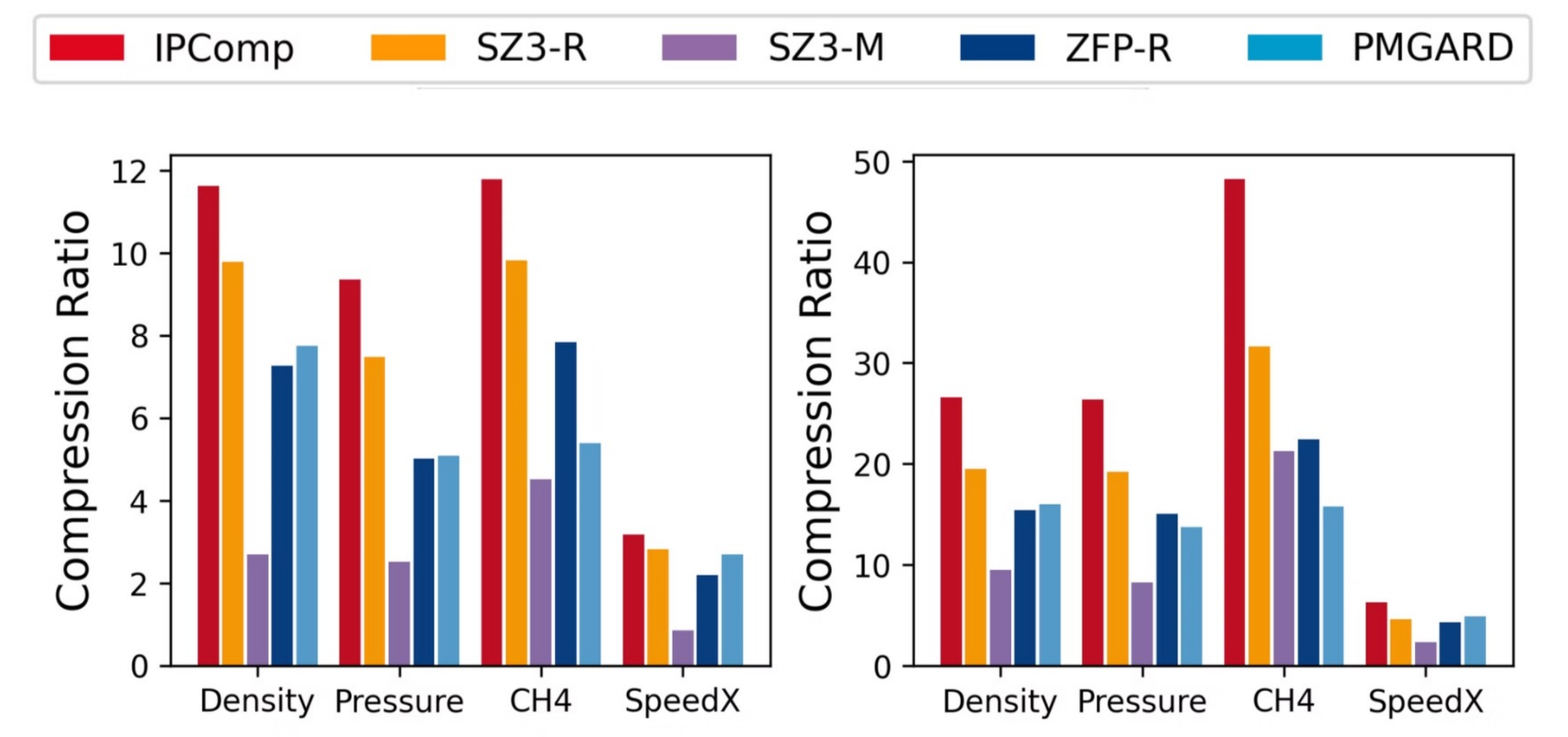
$$\max_{b_l, l \in \{1,2,\dots L\}} \sum_l SavedSize(l, b_l),$$

$$\text{subject to} \sum_l err(l, b_l) + eb \leq \text{E}.$$

- ➢ Can be solved using dynamic programming

Evaluation: IPComp

➢ **Compression Ratio = Original Data Size / Compressed Data Size**



(a) High precision setting ($eb = 1\text{e}{-9}$) (b) High ratio setting ($eb = 1\text{e}{-6}$)

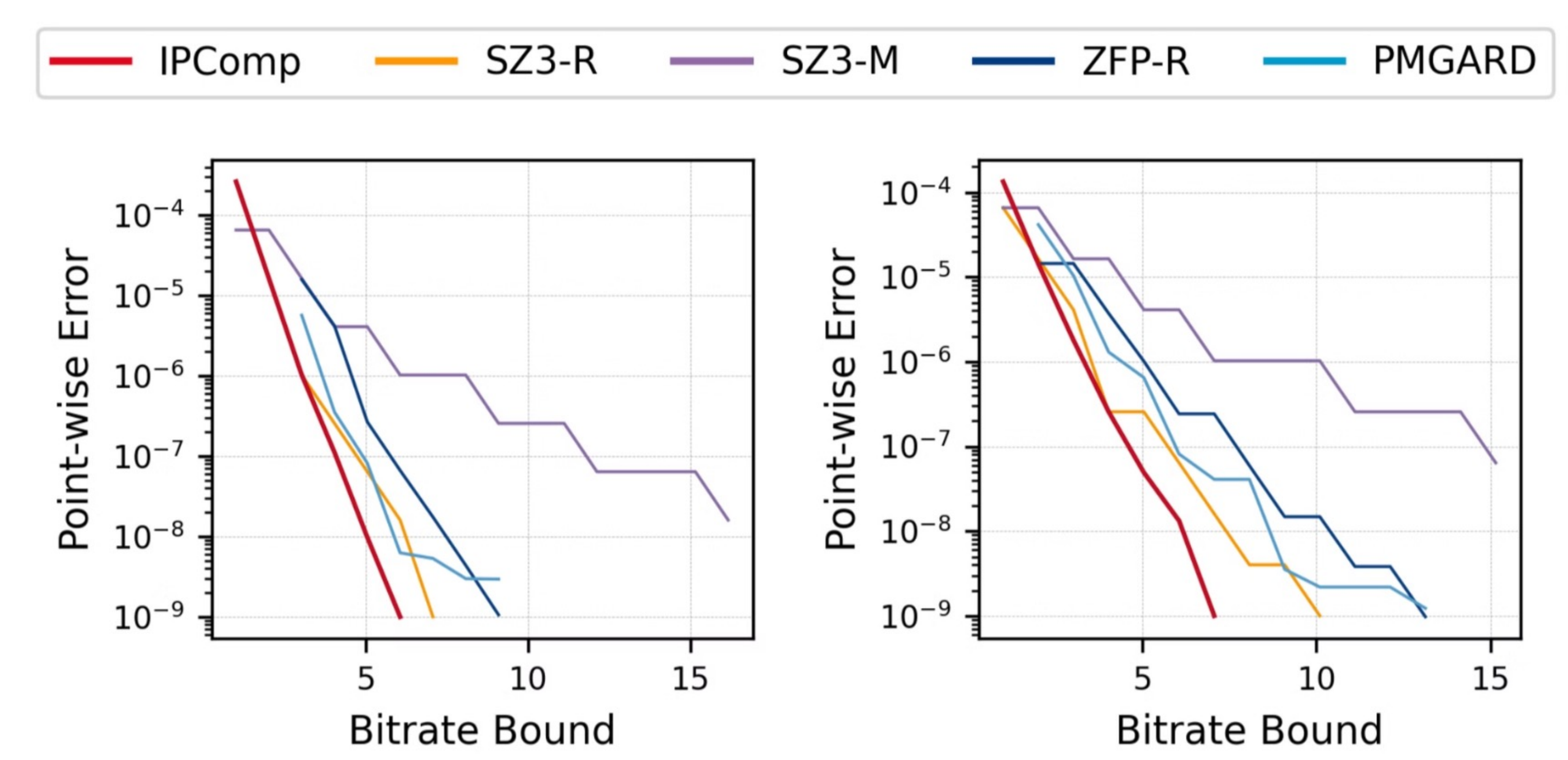
- ➢ SZ3-R: **residual compression** version of SZ3, stores **residuals**
- ➢ SZ3-M: the original version of SZ3, **stores multiple copies** of the data at different precision levels.
- ➢ ZFP-R: **residual compression** version of ZFP
- ➢ PMGARD: MGARD with support for progressive compression.

➢ **High Compressibility**

Evaluation: IPComp

- **Data Retrieval Efficiency**

Fixed Rate Mode



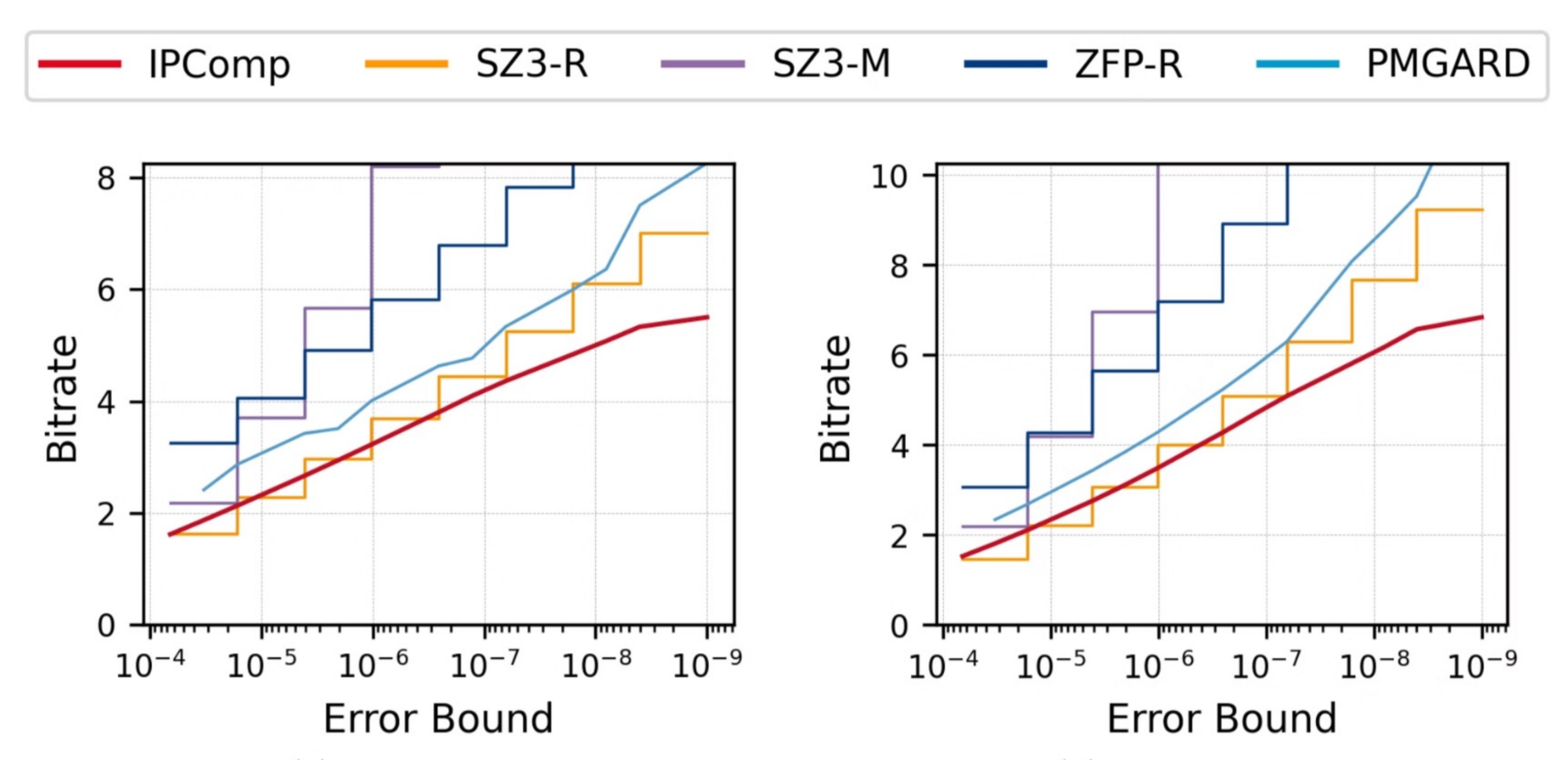
- Evaluate **Data Retrieval Efficiency** by incrementally loading data at increasing precision levels
- i.e., gradually tightening the bitrate bound from low to high
- and measuring the resulting distortion between the reconstructed and original data.

- **High Data Retrieval Efficiency**

Evaluation: IPComp

- **Data Retrieval Efficiency**

Error Bound Mode

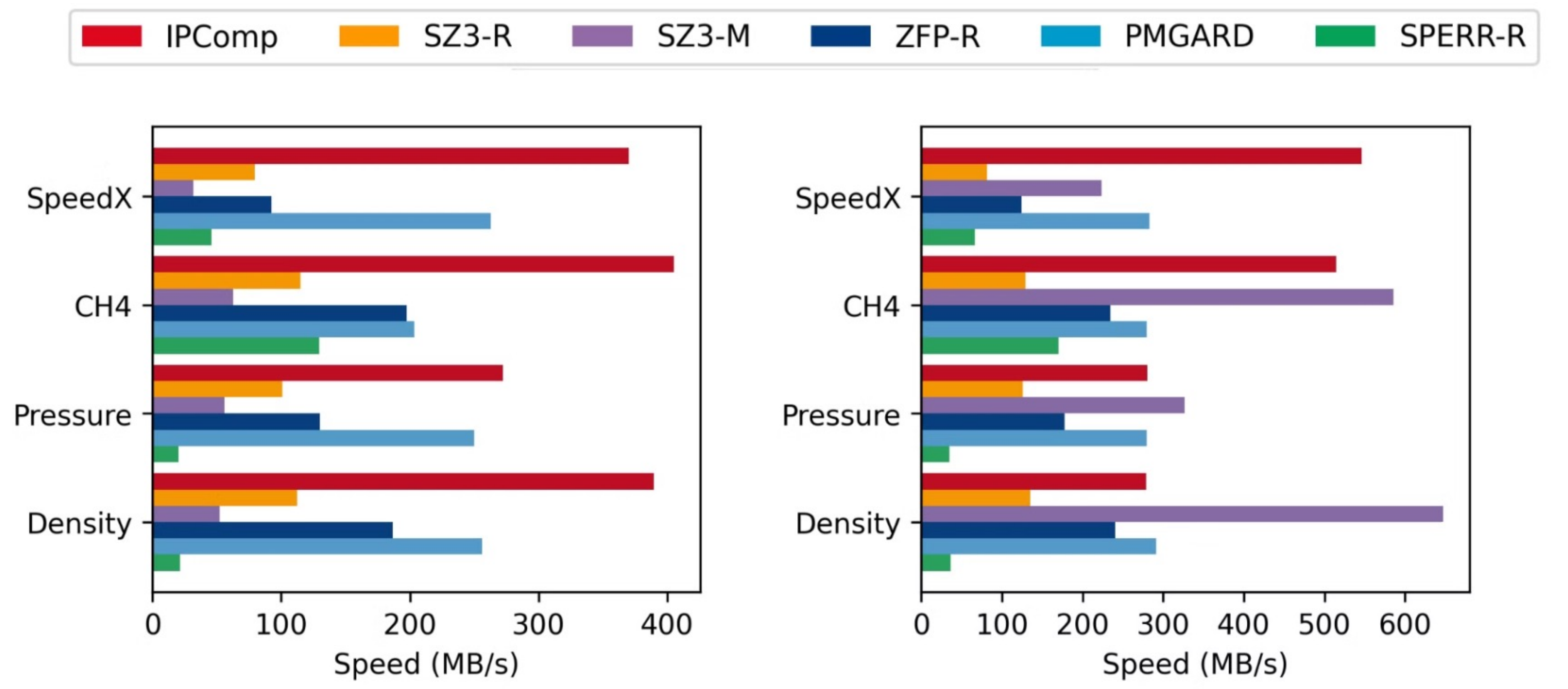


- progressively **tightening the error bound**
- and measuring the corresponding amount of loaded data

- **High Data Retrieval Efficiency**

Evaluation: IPComp

➢ Performance



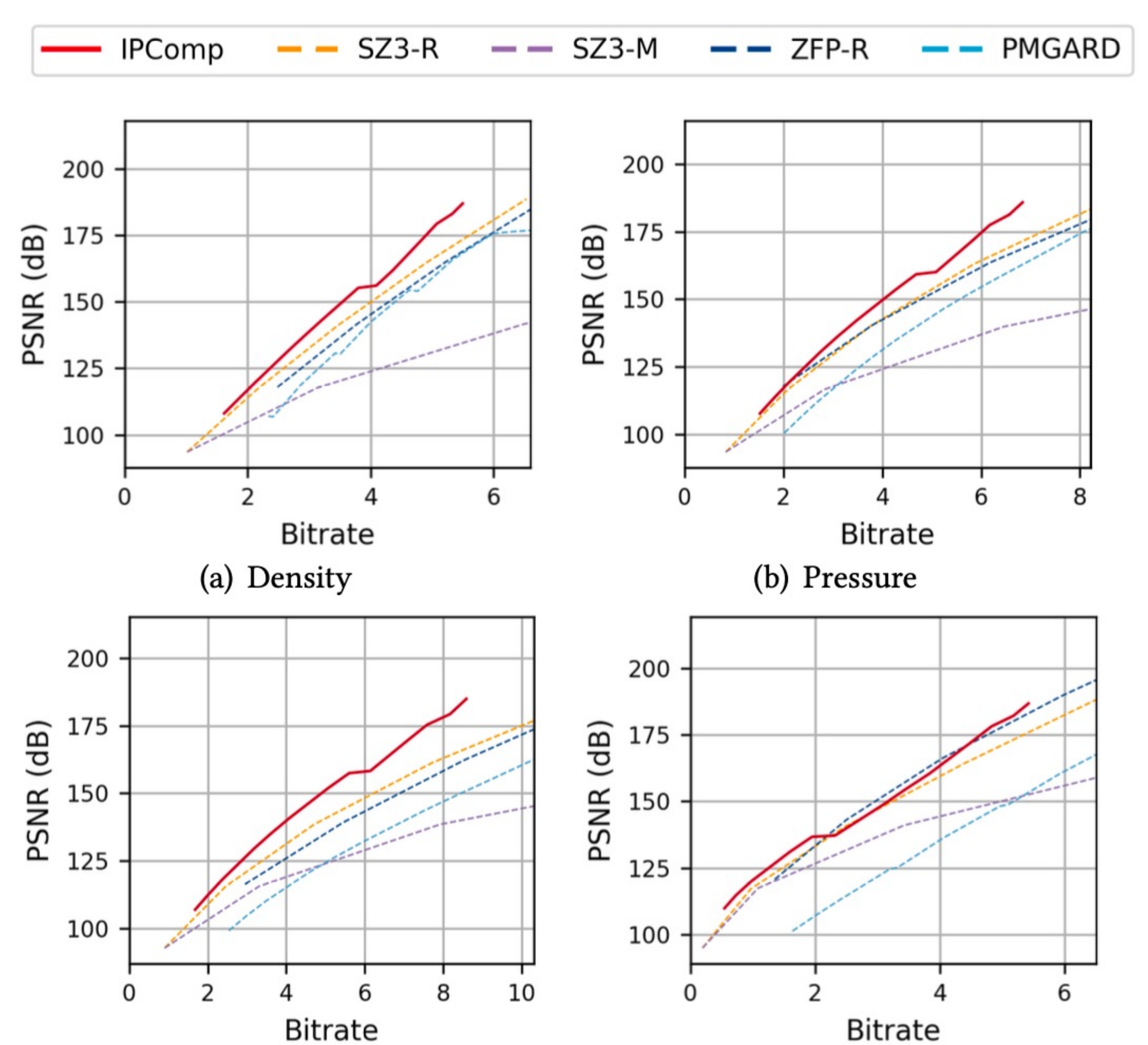
(a) Compression (b) Decompression

➢ residual-based approaches (-R) require multiple rounds of residual compression and decompression, making them slower

➢ **High Speed**

Evaluation: IPComp

Reconstruction Quality



(a) Density (b) Pressure

(c) VelocityX (d) CH4

Rate-Distortion Curve

- The deviation between the reconstructed data and the original data under a given bitrate
- use **PSNR (Peak Signal-to-Noise Ratio)** as the distortion metric
- Higher PSNR: better reconstruction quality, higher fidelity, and greater accuracy

- **Highest Reconstruction Accuracy**

Evaluation: IPComp

- **Reconstruction Quality**

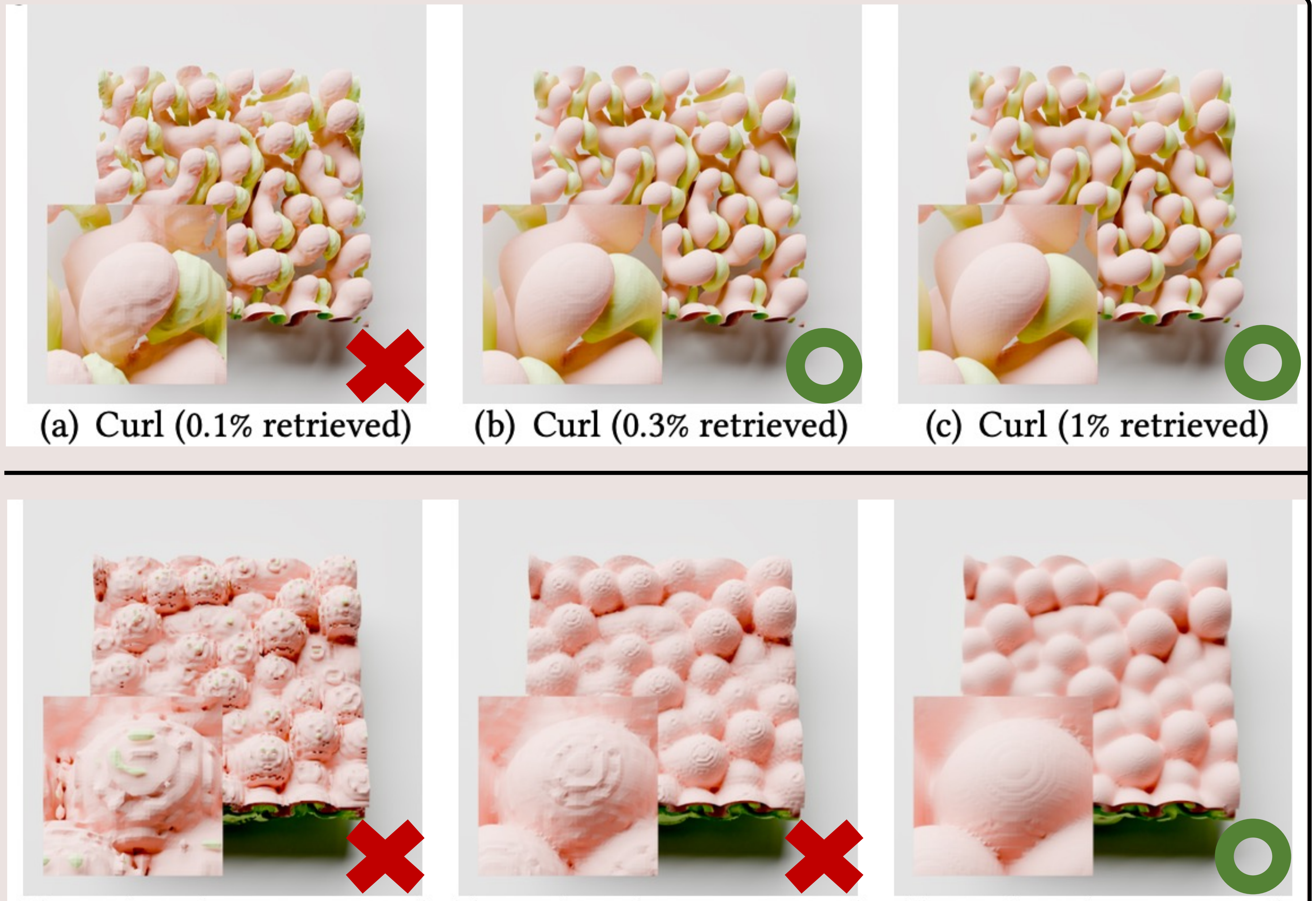
Visualization

- **Curl** and **Laplace** operator
- Recovering **0.1%, 0.3%, 1.0%** of the data

Curl: $\geq 0.3\%$

Laplace: $\geq 1.0\%$

- **The Necessity of Progressive Retrieval**



(a) Curl (0.1% retrieved) (b) Curl (0.3% retrieved) (c) Curl (1% retrieved)

(d) Laplace (0.1% retrieved) (e) Laplace (0.3% retrieved) (f) Laplace (1% retrieved)

Usage of IPComp

1 Error Bound Mode

Users specify required precision; optimizer minimizes data loaded while keeping error within bounds

2 Fixed Rate/Size Mode

Users specify maximum bitrate; optimizer minimizes error while staying within size limits

- The Optimized Data Loader also supports solving the inverse problem — minimizing the error under a given bitrate — using a method similar to the error-bound mode.

Code is available at
https://github.com/szcompressor/IPComp

Command-line Syntax:

```
IPComp -dataType[f/d] -dim_num … -bound_mode -bound_num …
```

Error-Bounded Progressive Compression

```
./src/IPComp density.d64 -d -3 256 384 384 -error -3 1e-3 1e-4 1e-5
```

progressive error bounds at 1e-3, 1e-4, and 1e-5.

Bitrate-Bounded Progressive Compression

```
./src/IPComp density.d64 -d -3 256 384 384 -bitrate -3 1.0 2.0 3.0
```

bitrate constraints of 1.0, 2.0 and 3.0 bits per value.

Q&A