Serf: Streaming Error-Bounded Floating-Point Compression @SIGMOD 2025

Ruiyuan Li [1,2], Zechao Chen [1,2], Ruyun Li [1,2], Xiaolong Xu [1,2], Guangchao Yang [1,2], Chao Chen [1], Jie Bao [3], Yu Zheng [3]

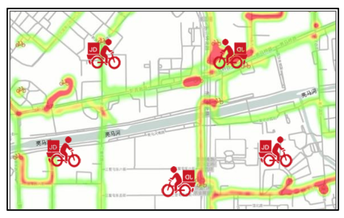
[1] *Chongqing University, China* [2] *Start Lab, China* [3] *JD iCity, JD Technology, China*

Background

➢ **Explosion of Floating-Point Time Series in a Streaming Way**

10k+ records/s

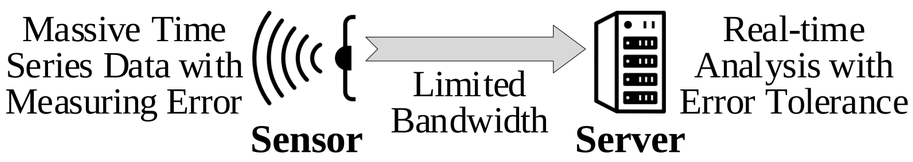
500GB data/flight



1TB trajectories/day

➢ **Scenarios of Streaming and Error-Bounded Compression**

- **Limited Bandwidth**: e.g., Zigbee 20~250 kbit/s
- **Error Tolerance**: Measuring Errors in Sensors+Unnecessary Exact Readings
- **Real-Time Requirement**: e.g., Taxi Dispatching, School Bus Monitoring



Motivation

➢ **Existing Floating-Point Compression Methods**

	Batched	Streaming
Lossless	LZ77, Zstd, Snappy, ALP, Xz, ···	Gorilla, Elf, Chimp, LZ4, ···
Lossy	SZ, SZ_ADT, Machete, Buff, ...	?

Cons:
- High Latency
- Memory Consumption
- Poor Compression Ratio

Pros:
- Low Latency

Cons:
- Poor Compression Ratio

Cons:
- High Latency
- Memory Consumption

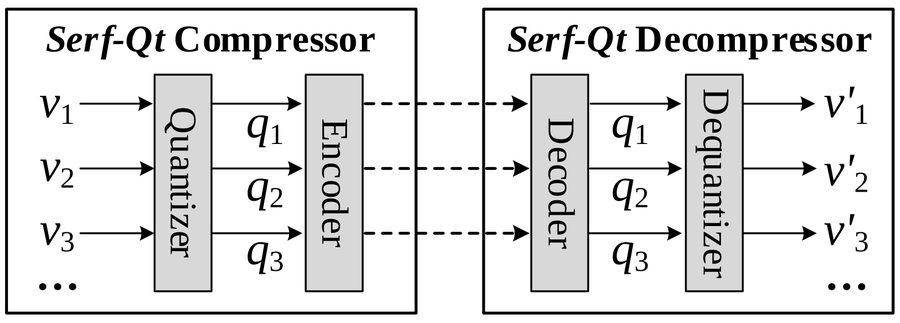
No Existing Streaming Lossy Floating-Point Compression Methods!

➢ **Challenges of Streaming and Error-Bounded Compression**

- How to achieve an excellent compression ratio with error bound guarantee in the streaming environments?
- How to ensure high efficiency under the limited resources in sensors?

Quantization-Based *Serf-Qt*

➢ **Framework of *Serf-Qt***



➢ **Quantizer**

$$q_i = \left[\frac{v_i - p_i}{2 \times \epsilon_a}\right]$$

➢ **Dequantizer**

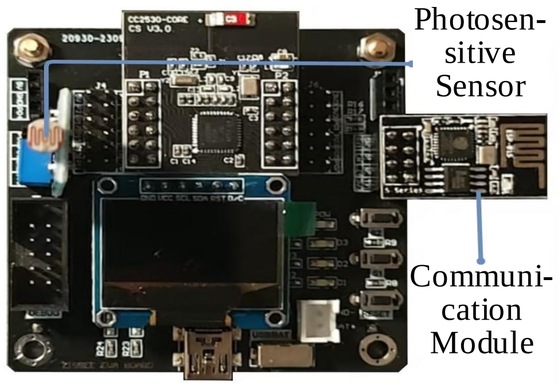
$$v'_i = 2 \times \epsilon_a \times q_i + p_i$$

Fast, But Without Relative Error Bound Support.

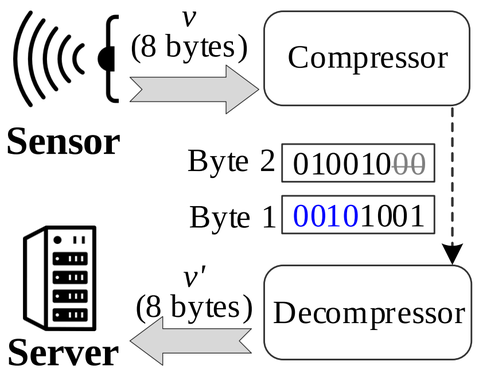
➢ **Encoder**: $code_i = \text{EliasGammaEn}(\text{ZigzagEn}(q_i)+1)$

➢ **Decoder**: $q_i = \text{ZigzagDe}(\text{EliasGammaDe}(code_i)-1)$

Transmission Prototype



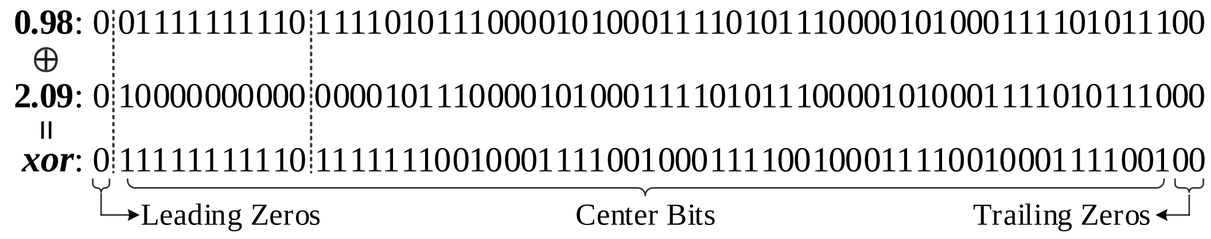
(a) Development Board: Dofly CC2530



(b) Transmission Framework

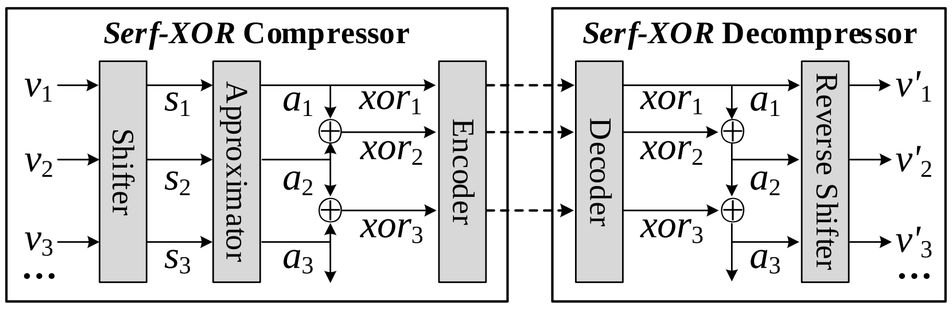
XORing-Based *Serf-XOR*

➢ **Motivation**: Few Leading Zeros and Trailing Zeros in XORed Values



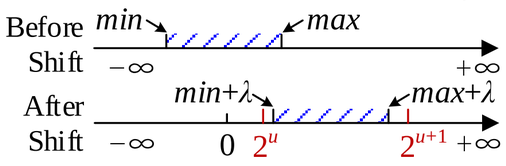
➢ **Framework of *Serf-XOR***

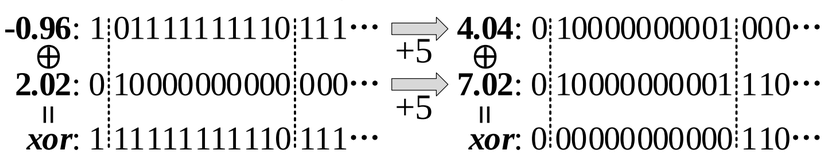
Support Both of Absolute and Relative Error Bounds



➢ **Shifter**: To Increase #Leading Zeros

$$s_i = v_i + \lambda, \lambda = 2^u - \lfloor min \rfloor, u = \lceil \log_2(\lfloor max \rfloor - \lfloor min \rfloor + 1) \rceil$$

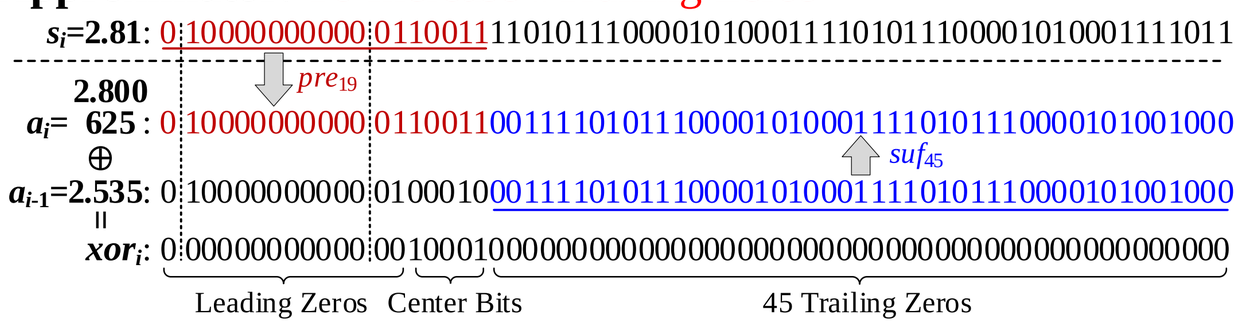




➢ **Reverse Shifter**

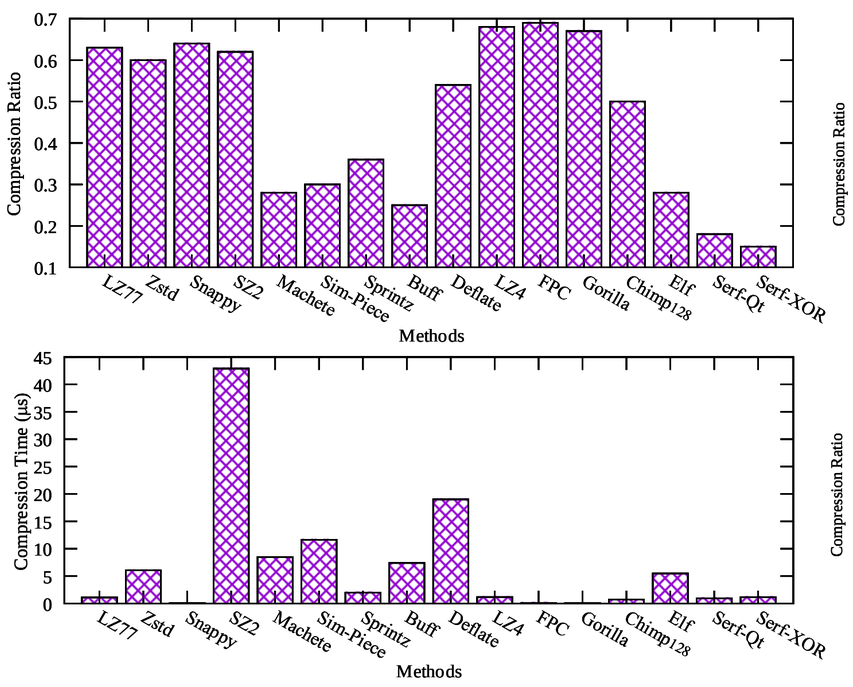
$$v'_i = a_i - \lambda$$

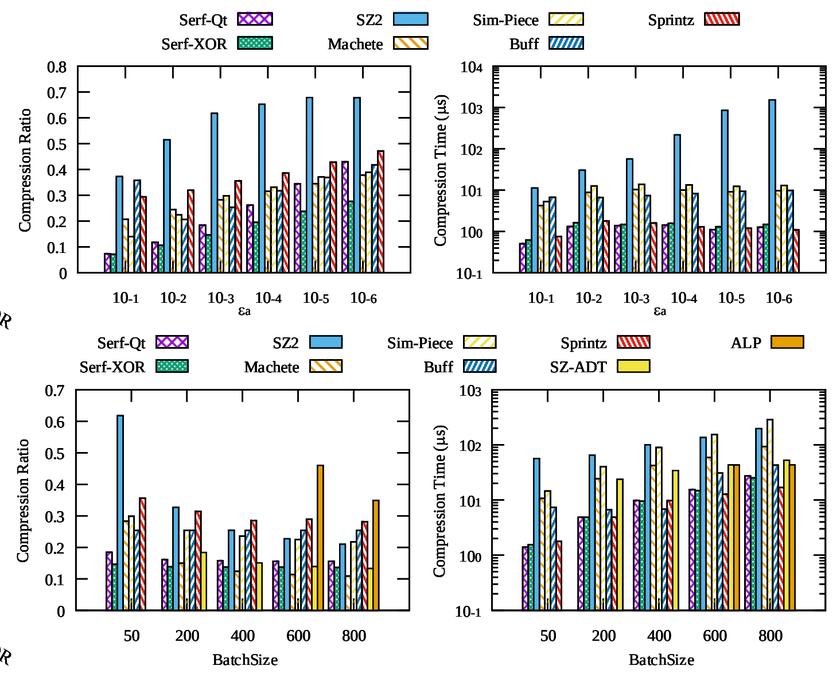
➢ **Approximator**: To Increase #Trailing Zeros

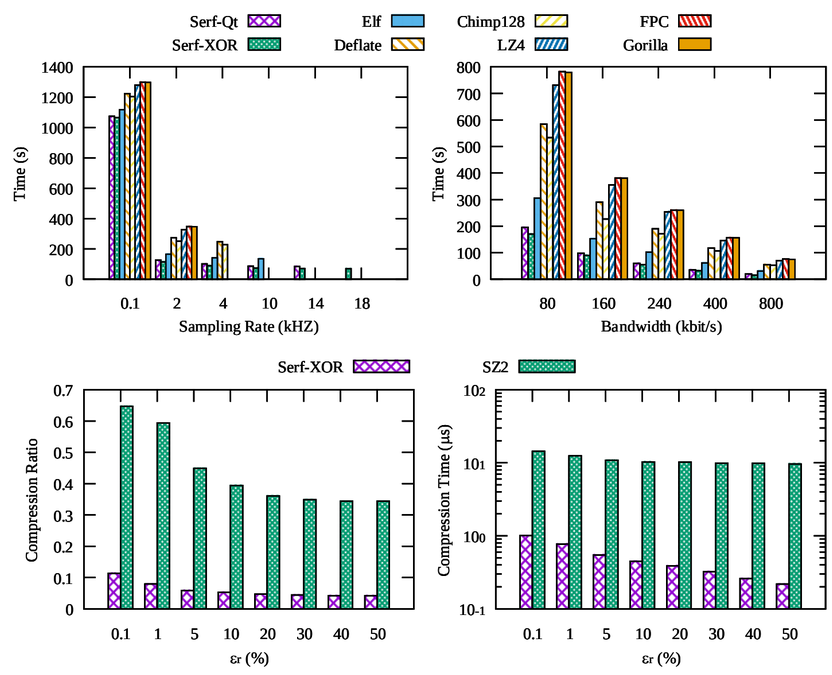


➢ **Encoder & Decoder**: Zheng Li, Ruiyuan Li*, et al. Adaptive Encoding Strategies for Lossless Floating-Point Compression. IEEE Internet of Things Journal. (IoTJ 2025)

Experiments (13 Datasets, 17 Competitors. https://github.com/Spatio-Temporal-Lab/Serf)







Compression Ratio ⬇40% over Buff, Only 15.7% Time.
Compression Ratio ⬇16.7% over *Serf-Qt*, Similar Time.

Smaller ϵ_a, Advantage of *Serf-XOR* is More Obvious.
Serf Holds Potential Using Mini-Batched Compression.

Serf-XOR Performs the Best in Data Transmission.
Compression Ratio ⬇91.7% over SZ2, Only 7.4% Time.