自我介绍

- 基本信息：http://www.kangry.net/blog/ruiyuanli 或搜索“李瑞远”
  - 中国电子学会优博(2022)，陕西省计算机学会优博(2020)，中国专利优秀奖(2021)
  - 研究方向：大数据，数据库，时空数据管理与挖掘，分布式计算，城市计算

- 教育经历
  - 2009.09~2013.06　武汉大学　本科　计算机科学与技术
  - 2013.09~2016.06　武汉大学　硕士　计算机软件与理论
  - 2016.09~2020.08　西安电子科技大学　博士　计算机系统结构，导师：郑宇教授
- 工作经历
  - 2014.07~2018.01　微软亚洲研究院　实习研究员　导师：郑宇教授
  - 2018.01~2020.08　京东智能城市研究院　实习研究员　导师：郑宇教授
  - 2020.08~2021.10　京东智能城市研究院　研究员/机构负责人　下属人数：约30人
  - 2021.10~至今　重庆大学计算机学院　副教授、硕导

高校研究背景 + 企业落地经验

时空实验室公众号

Elf: Erasing-based Lossless Floating-Point Compression

Ruiyuan Li, Associate Professor

ruiyuan.li@cqu.edu.cn

Chongqing University Start Lab, Chongqing, China

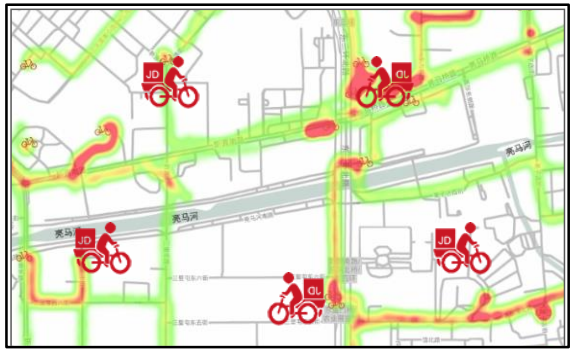
2023.08.23

Floating-Point Compression is Vital

- **Explosion** of Floating-Point Time Series Data in a **Streaming** Fashion

10k+ records/s

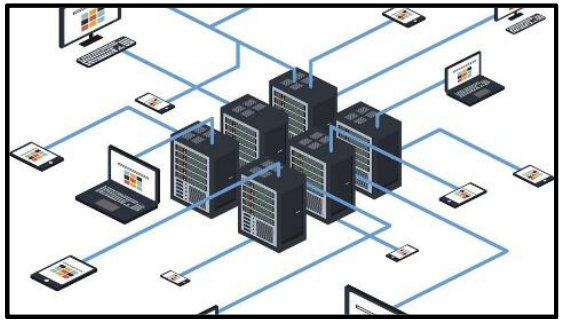
500GB data/flight



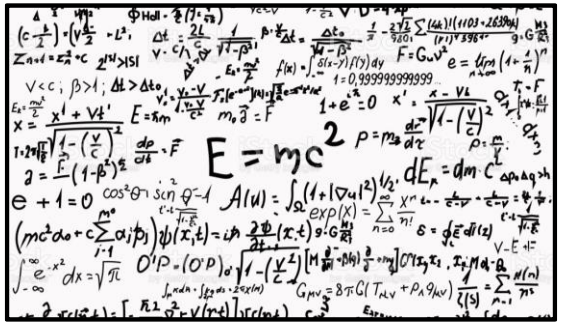
1T trajectories/day

- **Efficient**, **Compact** and **Lossless** Floating-Point Compression is Crucial
  - Reduce bandwidth/storage cost, improve transmission/query efficiency
  - Avoid information loss (e.g., small errors lead to big problems)

Network Transmission

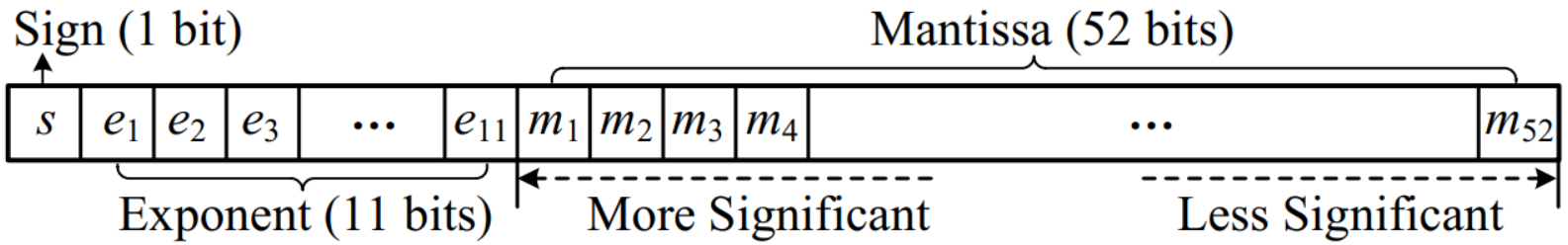


Data Management



Scientific Calculation

Complex Floating-Point Layout



IEEE 754 Double-Precision Floating-Point Layout

Focus on these two types

- **Normal Numbers:** $\exists\ i, j \in [1, 11], e_i = 0$ and $e_j = 1$
- **Zero:** $\forall\ i \in [1, 11], e_i = 0$ and $\forall\ j \in [1, 52], m_j = 0$

$$v = (-1)^s \times 2^{e-1023} \times (1.m_1 m_2 \ldots m_{52})_2$$

$$= (-1)^s \times 2^{e-1023} \times (1 + \sum_{i=1}^{52} m_i \times 2^{-i})$$

Normal Numbers

- **Infinity:** $\forall\ i \in [1, 11], e_i = 1$ and $\forall\ j \in [1, 52], m_j = 0$
- **NaN:** $\forall\ i \in [1, 11], e_i = 1$ and $\exists\ j \in [1, 52], m_j = 1$
- **Subnormal Number:** $\forall\ i \in [1, 11], e_i = 0$ and $\exists\ j \in [1, 52], m_j = 1$

Can be easily extended to these three Types

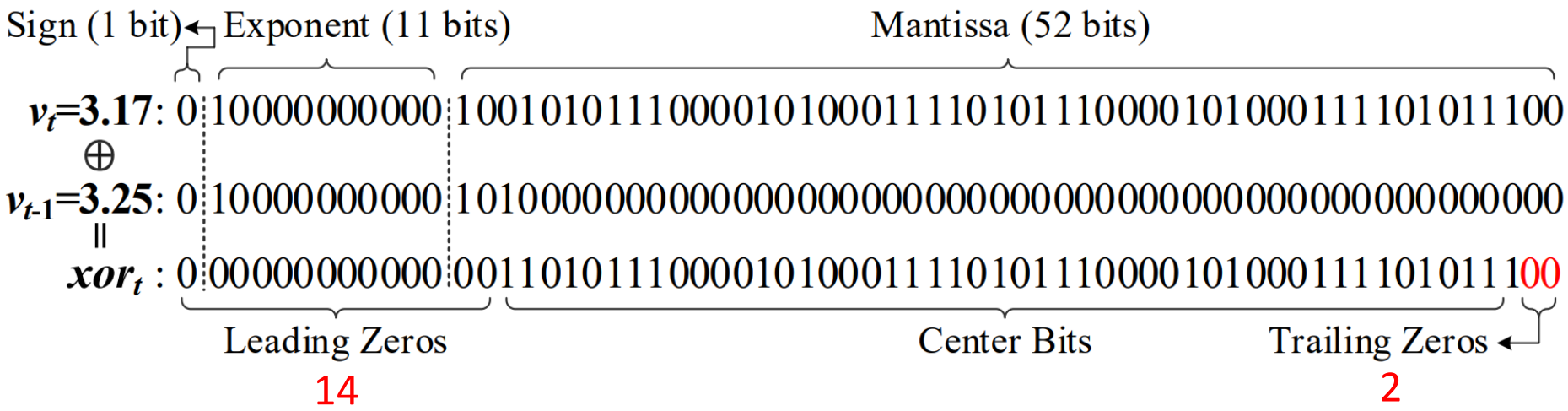
$$v = (-1)^s \times 2^{-1022} \times (0.m_1 m_2 \ldots m_{52})_2$$

$$= (-1)^s \times 2^{-1022} \times \sum_{i=1}^{52} m_i \times 2^{-i}$$

Subnormal Numbers

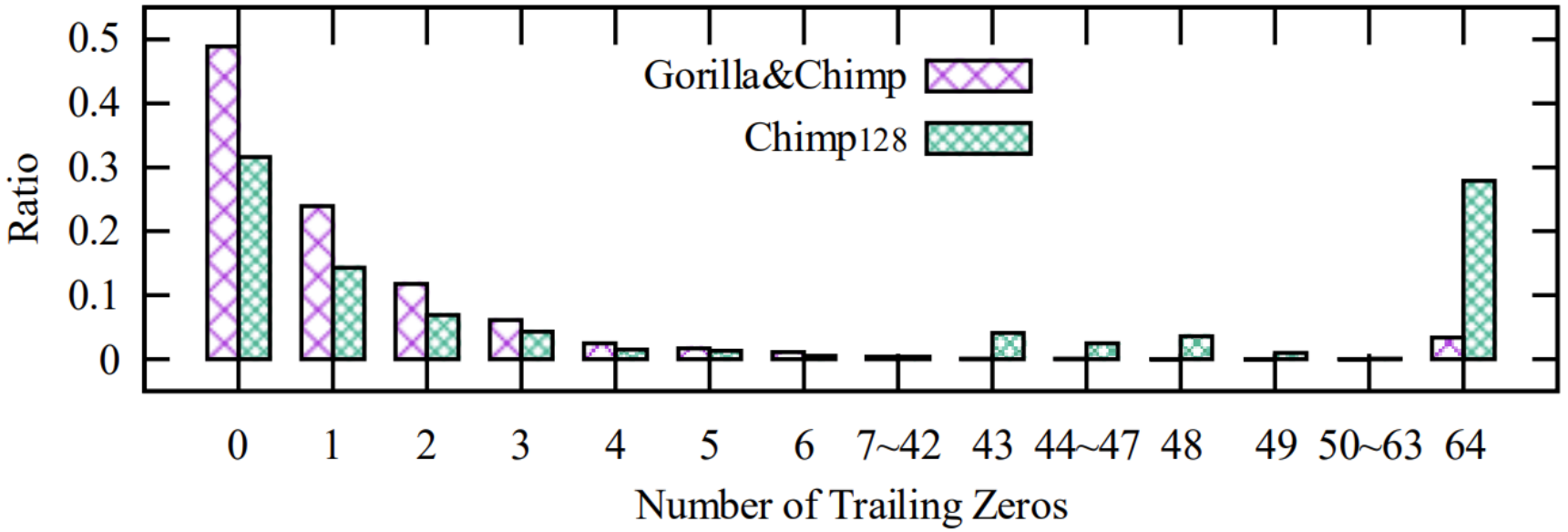
Shortcomings of Existing Solutions

- **Existing Solutions**
  - General-purpose methods, e.g., Gzip, Zstd, Xz, Snappy ➡ Batched, Low Efficiency ☹
  - Lossy floating-point methods, e.g., ZFP, MDZ ➡ Loss Some Information ☹
  - Lossless floating-point methods, e.g., Gorilla, Chimp (VLDB 2015) (VLDB 2022) ➡ Unsatisfactory Effects ☹
- **Lossless XORing-Based Time Series Compression Method**
  - **Compression**: $v_t \oplus v_{t-1} = xor_t$, then encode xor_t (e.g., $\#lead + \#trail$ + Center Bits)
  - **Decompression**: decode xor_t, then $xor_t \oplus v_{t-1} = v_t$
  - v_t and v_{t-1} vary little, so $\#lead$ is large. But how about $\#trail$?



Shortcomings of Existing Solutions

- In Fact, Most xor_t Contain Very Few Trailing Zeros
  - Gorilla&Chimp ($xor_t = v_t \oplus v_{t-1}$): 95% less than 5 trailing zeros
  - $Chimp_{128}$ ($xor_t = v_t \oplus v_{t-k}$, $0 < k \leq 128$): 60% less than 5 trailing zeros



- Few Trailing Zeros Lead to Unsatisfactory Compression Ratio
- Increasing Trailing Zeros Plays a Fundamental Role in Enhancing Performance

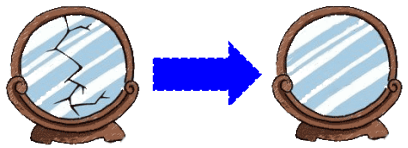
Intuition of Elf

- Erasing Last Few Bits (i.e., Setting Zeros) of Mantissa
- Restoring Mantissa with a Leave-Out Operation

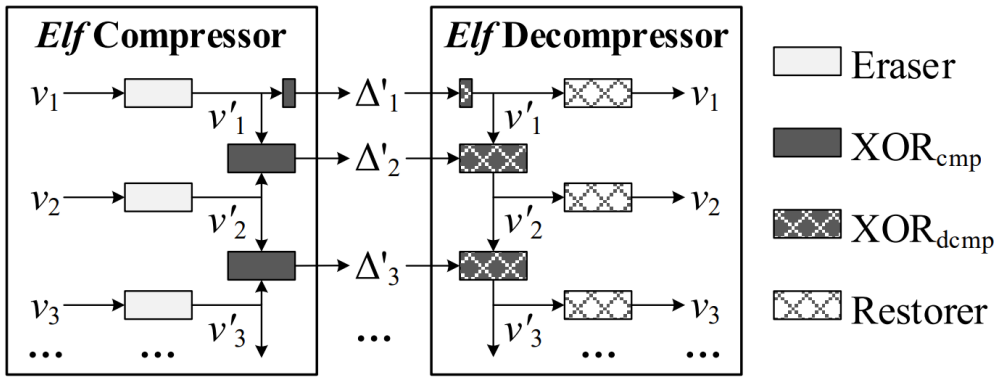
Compress	**3.17**: 0 10000000000 1001010111000010100011111010111000010100011111010111100 Erase the last 44 bits. ⇩ 3.1640625=3.17−δ, 0<δ<0.01 **3.164 0625**: 0 10000000000 1001010100000000000000000000000000000000000000 ⊕ **3.25**: 0 10000000000 1010000000000000000000000000000000000000000000000 = **Δ'** : 0 00000000000 0011010100000000000000000000000000000000000000000
Decompress	(1) Decode Δ'; (2) Δ' ⊕ 3.25 = 3.1640625; (3) v = 3.1640625 + δ, 0 < δ < 0.01, so v = 3.16~~40625~~ + 0.01 = 3.17

- Elf's Magic
  - Number of Trailing Zeros

    2 ➡ 44
  - A broken mirror join together



- Streaming Framework



Challenges

- How to **efficiently** determine the erased bits?

  Efficiency Requirement

- How to **compactly** encode the erased data?

  Compactness Requirement

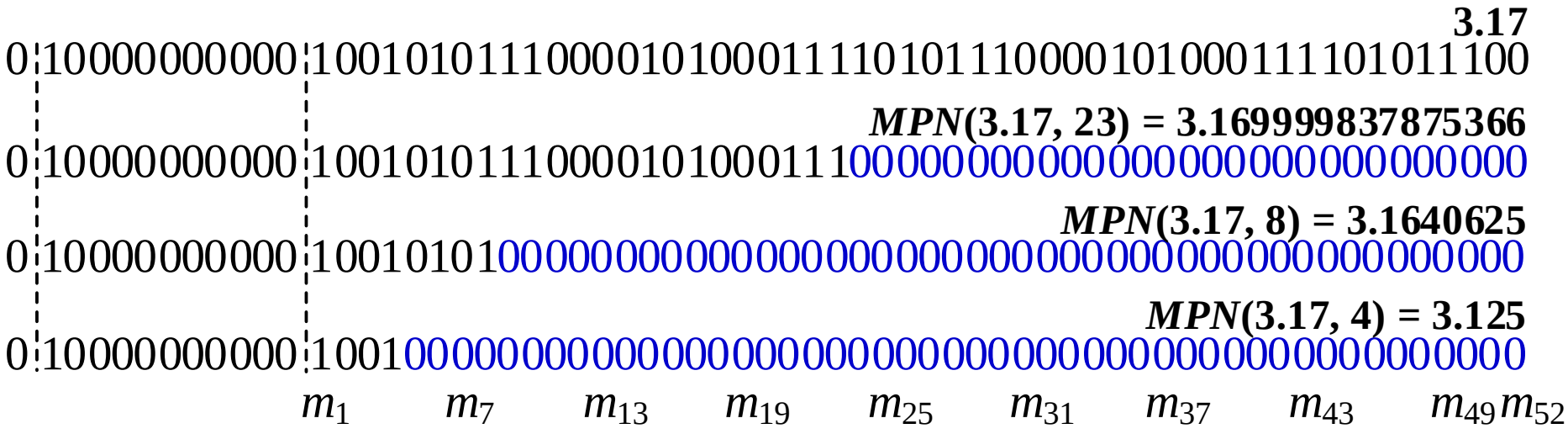
- How to **losslessly** restore the original data?

  Lossless Requirement

Challenge 1: Determine Erased Bits

- **Definition: Decimal Place Count $\alpha = DP(v)$**
  - $v = (\pm d_{h-1}d_{h-2}\ldots d_0.d_{-1}\ldots d_l)_{10}$, then $DP(v) = |l|$
  - Example: $DP(3.17) = 2$, $DP(-0.0317) = 4$, $DP(317.0) = 1$

- **Definition: Mantissa Prefix Number $MPN(v, n)$**
  - $v' = MPN(v, n)$: erasing (i.e., setting zeros) the bits after m_n in v



Challenge 1: Determine Erased Bits

- **To Ensure $xor_t = v_t \oplus v_{t-1}$ with Large *#trail*, Find $v' = MPN(v, n)$ Satisfying:**
  - v' contains trailing zeros as many as possible
  - $0 < \delta = v - v' < 10^{-\alpha}$, so we can restore the original data

Suppose $v > 0$

- **Basic Method**
  - Enumerate n from 52 to 1, until $\delta = v - v' \geq 10^{-\alpha}$
  - Shortcomings: time-consuming, $O(52)$ ☹
- **Elf Solution**
  - Theorem: erase the bits after $m_{g(\alpha)}$ directly, i.e., $v' = MPN(v, g(\alpha))$
  - $\alpha = DP(v)$, $g(\alpha) = \lceil \alpha \times log_2 10 \rceil + \lfloor log_2 |v| \rfloor$, $O(1)$ 🙂

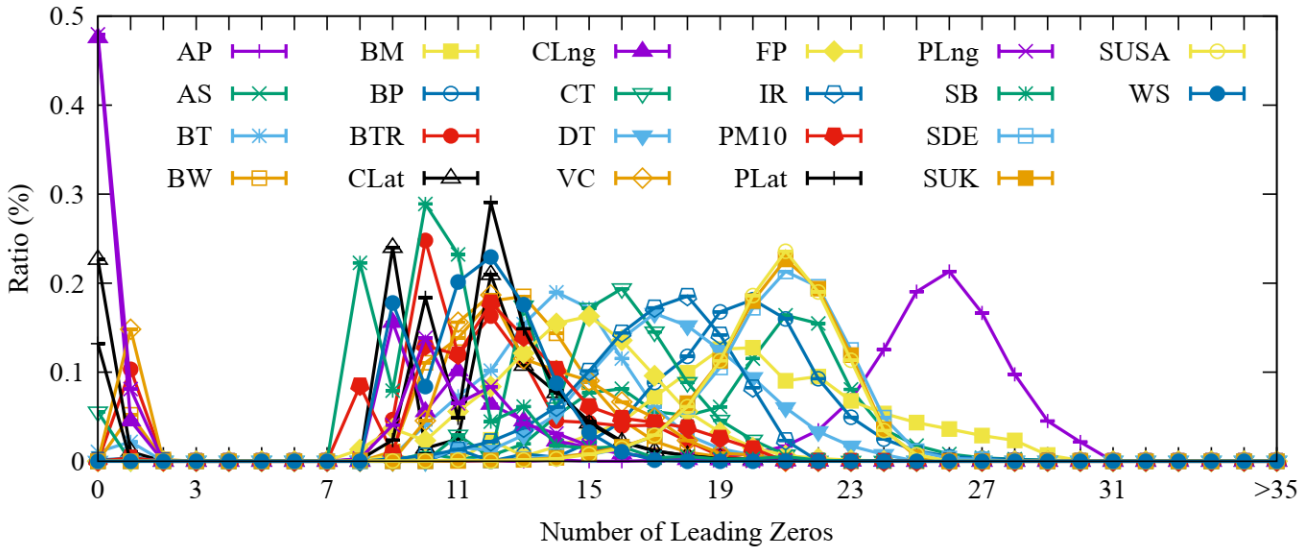
0|10000000000|1001010111000010100011110101110000101000111101011100 **3.17**

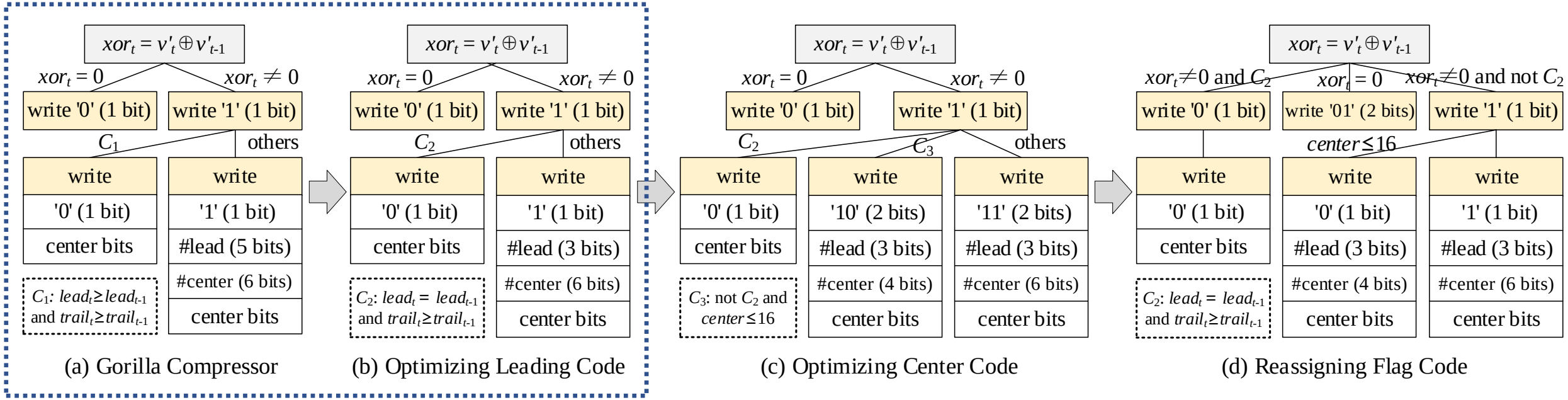
Since $g(\alpha) = \lceil 2 \times log_2 10 \rceil + \lfloor log_2 |3.17| \rfloor = 8$, *Elf* directly erases the bits after m_8

0|10000000000|10010101000000000000000000000000000000000000000000 **3.1640625**

Challenge 2: Encode Erased Data

- **Opt 1: Optimizing Leading Zeros**
  - *#lead* is severely unbalanced
  - Approximate *#lead* using 8 steps
    - *Rule* = (0, 8, 12, 16, 18, 22, 24), 3 bits
    - E.g., if *#lead* = 13, we regard it as 12
- **Opt 2: Optimizing Center Code**
- **Opt 3: Reassigning Flag Code**

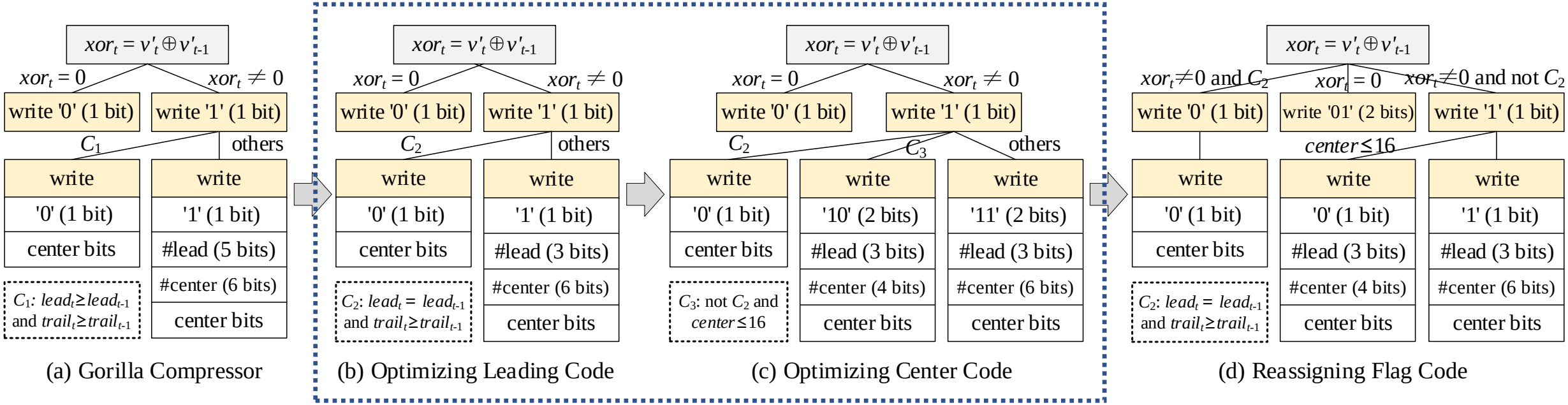




(a) Gorilla Compressor (b) Optimizing Leading Code (c) Optimizing Center Code (d) Reassigning Flag Code

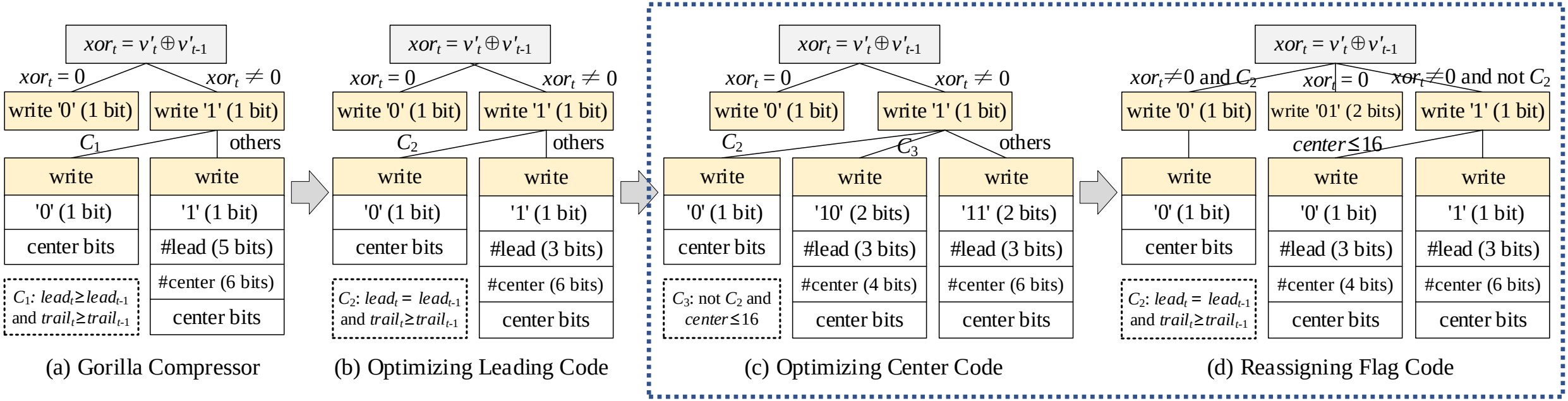
Challenge 2: Encode Erased Data

- Opt 1: Optimizing Leading Zeros
- Opt 2: Optimizing Center Code
  - xor_t has both many leading zeros and trailing zeros, so $\#center \leq 16$ in most cases
  - If $\#center \leq 16$, we use 4 bits + 1 flag bit for $\#center$
  - If $\#center > 16$, we use 6 bits + 1 flag bit for $\#center$
  - Considering the extra bit of flag, on average, we use less than 6 bits for $\#center$
- Opt 3: Reassigning Flag Code



Challenge 2: Encode Erased Data

- Opt 1: Optimizing Leading Zeros
- Opt 2: Optimizing Center Code
- Opt 3: Reassigning Flag Code
  - Identical consecutive values (i.e., xor_t=0) is infrequent, but it takes only 1 flag bit ☹
  - Other cases are frequent, but they take 2 or 3 flag bits ☹
  - To reduce overall flag bits, we reassign flag code where each case takes 2 bits ☺



Challenge 3: Restore Original Data

- **Three Steps**
  - **Step 1:** Decode xor_t (an inverse process of encoding)
  - **Step 2:** Recover $v'_t = xor_t \oplus v'_{t-1}$
  - **Step 3:** Restore v_t from v'_t, with a leave-out operation

$$v_t = d_{h-1}d_{h-2}...d_0.d_{-1}...d_{-\alpha}\cancel{d_{-(\alpha+1)}...d_l} + 10^{-\alpha}$$

e.g., $3.17 = 3.16\cancel{40625} + 10^{-2}$

$$v_t = RoundUp(v'_t, \alpha)$$

- **xor_t is decoded, and v'_{t-1} is already restored. How to obtain $\alpha = DP(v_t)$?**
- **Basic Method**
  - Since $v_{min} \approx 4.9 \times 10^{-324}$, store α with $\lceil \log_2 \alpha_{max} \rceil = \lceil \log_2 324 \rceil$ = 9 bits ☹
- **Elf Solution**
  - Store β (see next slide) instead of α with 4 bits, since α can be calculated by β 🙂

Challenge 3: Restore Original Data

- **Definition: Decimal Significand Count $\beta = DS(v)$**
  - $v = (\pm d_{h-1}d_{h-2}\ldots d_0.d_{-1}\ldots d_l)_{10}$, if $d_i = 0$ for $i \in (s+1, h-1)$ but $d_s \neq 0$, then $DS(v) = s - l + 1$
  - Example: $DS(3.17) = 3$, $DS(-0.0317) = 3$, $DP(317.0) = 4$, $DP(0.0) = undefined$
  - Theorem: $51 - \beta log_2 10 < \#\text{Erased Bits} < 53 - (\beta - 1) log_2 10$
  - $\beta \in \{1, \ldots, 17\}$, but we give up erasing if $\beta \geq 16$ (i.e., #Erased Bits < 4), so stored $\beta \in \{1, \ldots, 15\}$

- **Definition: Start Decimal Significand Position $SP(v)$**
  - $v = (\pm d_{h-1}d_{h-2}\ldots d_0.d_{-1}\ldots d_l)_{10}$, if $d_i = 0$ for $i \in (s+1, h-1)$ but $d_s \neq 0$, then $SP(v) = s$
  - Example: $SP(3.17) = 0$, $SP(-0.0317) = -2$, $SP(317.0) = 2$, $SP(0.0) = undefined$

- **Calculate α by β and $SP(v')$**

$$\alpha = \begin{cases} \beta - (SP(v') + 1) & v \neq 10^{-i}, i > 0 \\ \beta - (SP(v') + 2) & v = 10^{-i}, i > 0 \end{cases}$$

- **Store $\beta^* \in \{0, \ldots, 15\}$ with 4 bits** 

$$\beta^* = \begin{cases} 0 & v = 10^{-i}, i > 0 \\ \beta & others \end{cases}$$

Corner Cases

$v = 0.1$, $\alpha = DP(v) = 1$, $g(\alpha) = 0$, $\beta = DS(v) = 1$		$v' = 0.0625$, $\delta = v - v' = 0.0375$
0 01111111011 ~~1001100110011001100110011001100110011001100110011010~~	⇨	0 01111111011 0000000000000000000000000000000000000000000000000000

(a) Example of Erasing for $v = 10^{-i}$, $i > 0$. Set $\beta^* = 0$

$v = 3.141592653589792$, $\alpha = DP(v) = 15$, $g(\alpha) = 51$, $\beta = DS(v) = 16$		$v' = 3.1415926535897913$, $\delta = v - v' = 0.0000000000000007$
0 10000000000 1001001000011111101101010100010001000010110100010101~~1~~	⇨	0 10000000000 10010010000111111011010101000100010000101101000101010

(b) Example of Invalid Erasing When $\beta \geq 16$. Do Not Perform *Elf* Erasing

$v = 0.75$, $\alpha = DP(v) = 2$, $g(\alpha) = 6$, $\beta = DS(v) = 2$		$v' = 0.75$, $\delta = v - v' = 0$
0 01111111110 100000~~0000000000000000000000000000000000000000000000~~	⇨	0 01111111110 100000000000000000000000000000000000000000000000000000

(c) Example of Invalid Erasing When $\delta = 0$. Do Not Perform *Elf* Erasing

$v = 0.0$		$v' = 0.0$
0 00000000000 0000000000000000000000000000000000000000000000000000	⇨	0 00000000000 0000000000000000000000000000000000000000000000000000

(d) $v = 0$. Do Not Perform *Elf* Erasing

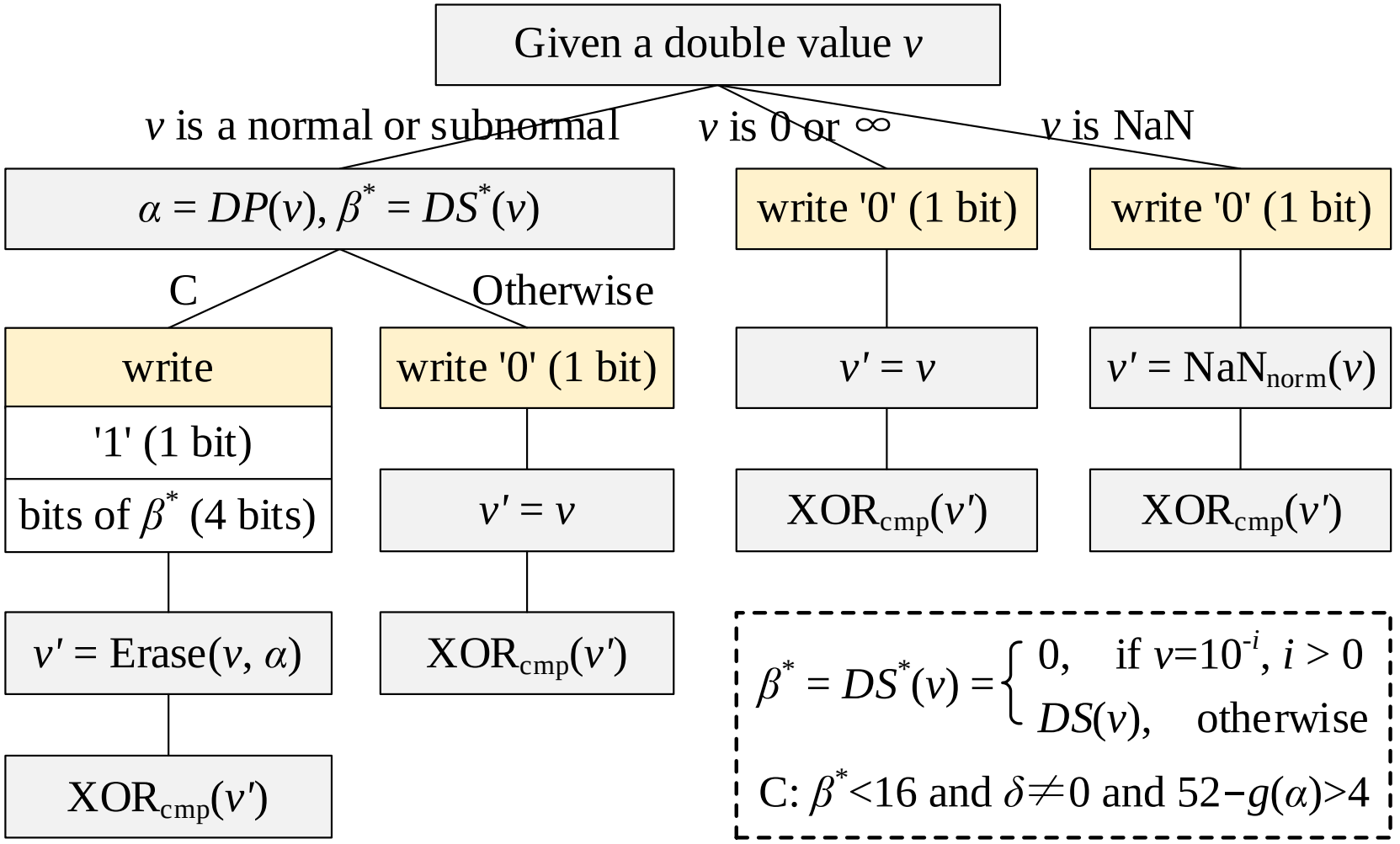
$v = \infty$		$v' = \infty$
0 11111111111 0000000000000000000000000000000000000000000000000000	⇨	0 11111111111 0000000000000000000000000000000000000000000000000000

(e) $v = \infty$. Do Not Perform *Elf* Erasing

v = NaN		v' = NaN_{norm}
0 11111111111 0000000000000000110000001110000000000000000000000000	⇨	0 11111111111 1000000000000000000000000000000000000000000000000000

(f) v = NaN. Set v' = NaN_{norm}

Overall Erasing Strategy



Experiments

- **22 Datasets**
  - 14 Time Series
  - 8 Non Time Series

		Dataset	#Records	β	Time Span
Time Series	Small β	City-temp (CT)	2,905,887	3	25 years
		IR-bio-temp (IR)	380,817,839	3	7 years
		Wind-speed (WS)	199,570,396	2	6 years
		PM10-dust (PM10)	222,911	3	5 years
	Medium β	Stocks-UK (SUK)	115,146,731	5	1 year
		Stocks-USA (SUSA)	374,428,996	4	1 year
		Stocks-DE (SDE)	45,403,710	6	1 year
		Dewpoint-temp (DT)	5,413,914	4	3 years
		Air-pressure (AP)	137,721,453	7	6 years
		Basel-wind (BW)	124,079	8	14 years
		Basel-temp (BT)	124,079	9	14 years
		Bitcoin-price (BP)	2,741	9	1 month
		Bird-migration (BM)	17,964	7	1 year
	Large β	Air-sensor (AS)	8,664	17	1 hour
Non Time Series	Small β	Food-price (FP)	2,050,638	3	-
		Vehicle-charge (VC)	3,395	3	-
	Medium β	Blockchain-tr (BTR)	231,031	5	-
		SD-bench (SB)	8,927	4	-
		City-lat (CLat)	41,001	6	-
		City-lon (CLon)	41,001	7	-
	Large β	POI-lat (PLat)	424,205	16	-
		POI-lon (PLon)	424,205	16	-

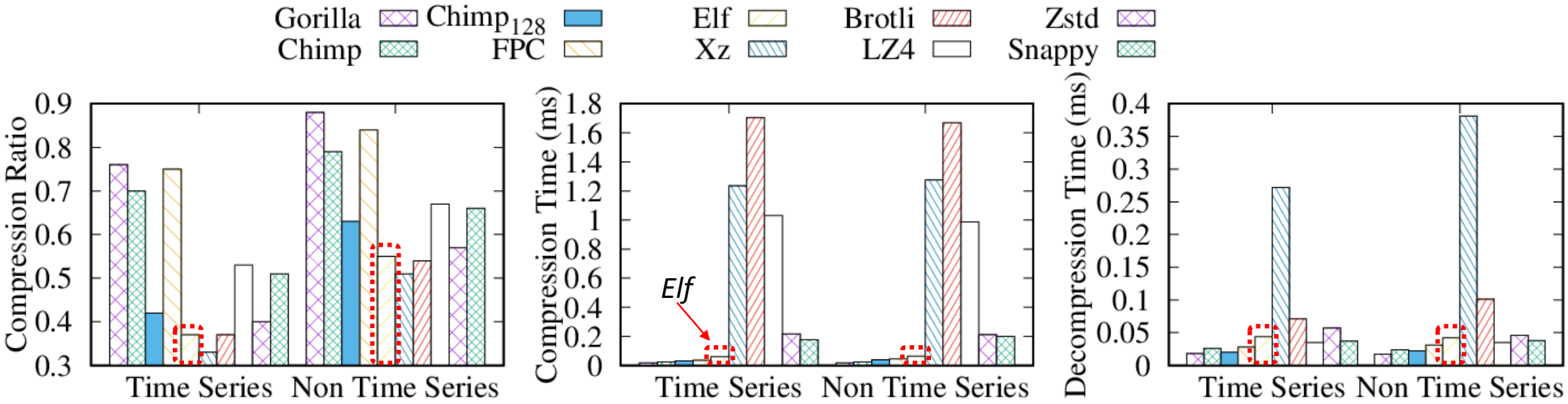
- **12 Baselines**
  - 4 lossless compression methods
    - Gorilla (VLDB 2015), Chimp (VLDB 2022), $Chimp_{128}$ (VLDB 2022), FPC (TC 2008)
  - 5 general compression methods
    - Xz, Brotili, LZ4, Zstd, Snappy
  - 3 variants
    - Gorilla + Eraser, Chimp + Eraser, Chimp128 + Eraser
- **3 Metrics**
  - Compression Ratio
    - Compressed Szie / Original Size
  - Compression Time
  - Decompression Time
- **Programing Language**
  - Java 1.8

The Smaller, The Better

Experiments

❑ Overall Comparison with Baselines

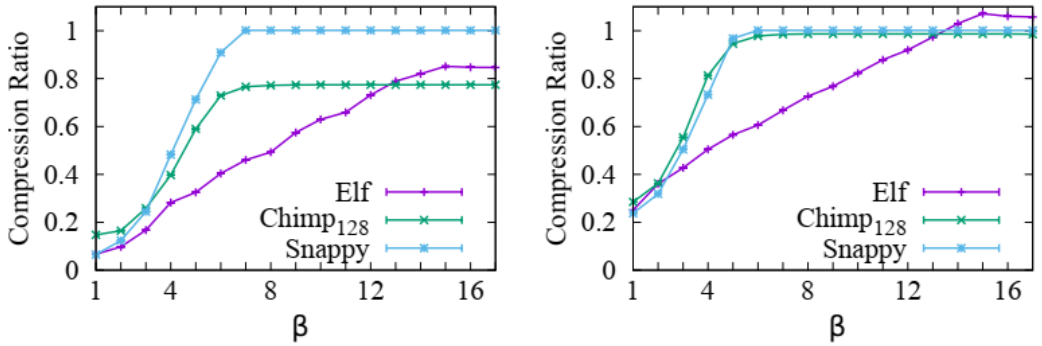
- ❑ Compared with $Chimp_{128}$ (best streaming competitor)
  - ❑ Elf improves 12.4% compression ratio, while with similar time
- ❑ Compared with Xz (best general competitor)
  - ❑ Elf has a competitive compression ratio, but only takes 4.84% compression time



Experiments

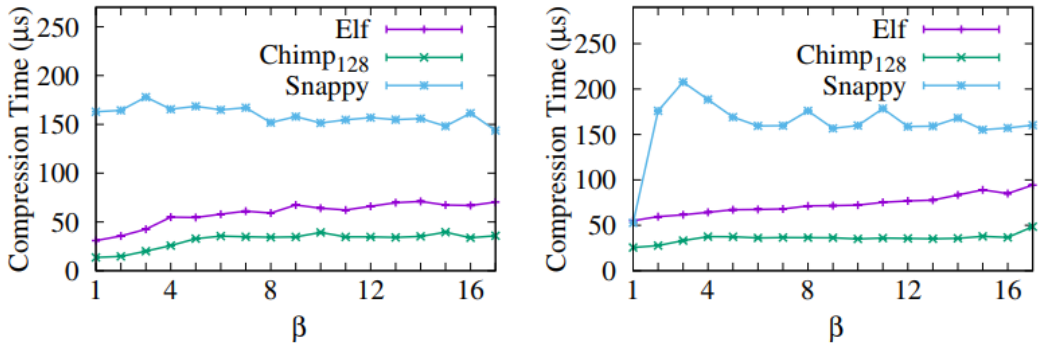
❑ Performance with Different β

- ❑ If β < 13, Elf always performs the best
- ❑ If β = 6, ⬆ 36.6% over Chimp128
- ❑ If β = 6, ⬆ 48.3% over Snappy



(a) Compression Ratio in AS.

(b) Compression Ratio in PLon.

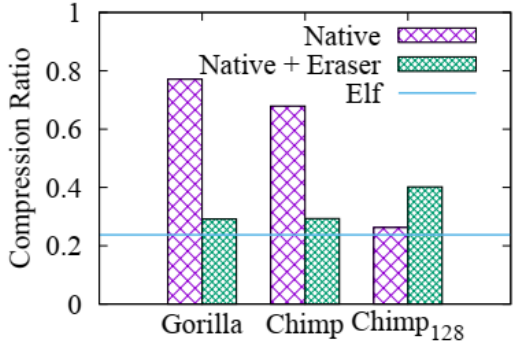


(c) Compression Time in AS.

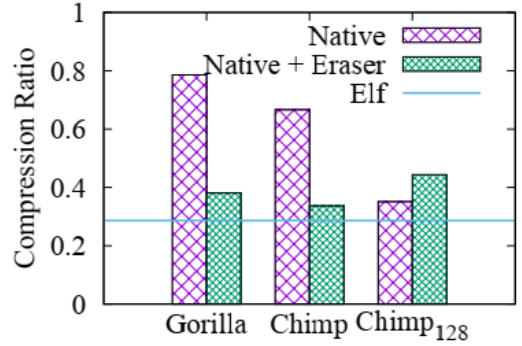
(d) Compression Time in PLon.

❑ Effectiveness of Erasing & Encoding

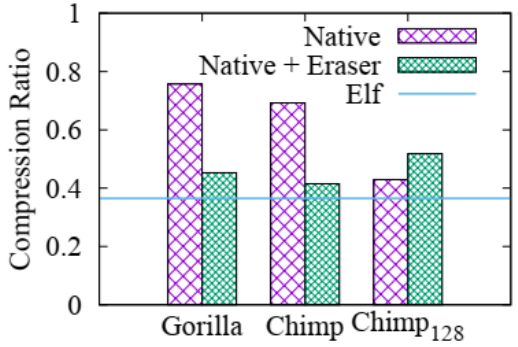
- ❑ Erasing ⬆ 56.5%~62.2% on Gorilla
- ❑ Erasing ⬆ 49.5%~51.6% on Chimp
- ❑ Encoding ⬆ 8.7%~33.3% on variants



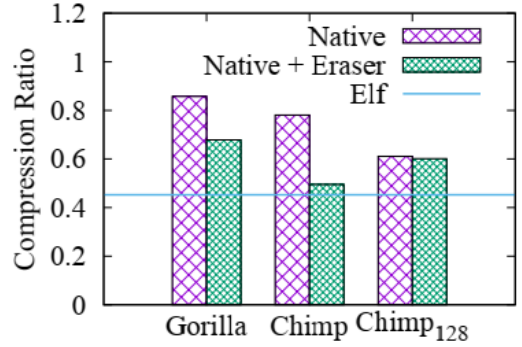
(a) Time Series with Small β.



(b) Non Time Series with Small β.



(c) Time Series with Medium β.



(d) Non Time Series with Medium β.

Discussion

- **Complexity: Both $O(1)$**
  - Efficient bitwise operation
  - Store previous value only
- **A Possible Variant**
  - Elf_k: $0 < \delta = v - v' < k \times 10^{-\alpha}$
  - Theorem: Elf_1 is better than Elf_k
- **Performance ⬆ in Transmission**
  - If bandwidth < 60M pbs, Elf is better than Chimp128 ($O(33K)$)
- **Can We Improve Elf Further?**

From: Microsoft CMT > Details

PVLDB **Reproducibility submission #1280 - Review Complete**

2023-07-19 04:47:55

The reproducibility process for your submission #1280 entitled "Elf: Erasing-based Lossless Floating-Point Compression(Reproducibility)" is now complete.

The Reproducibility Committee was able to reproduce your submission. Your paper will be highlighted on the VLDB Reproducibility Site and will be considered for the VLDB 2022 "Most Reproducible Paper" award.

Thank you for participating in the VLDB reproducibility process. Your efforts help to ensure that the VLDB community remains sustainable, vibrant, and trustworthy.

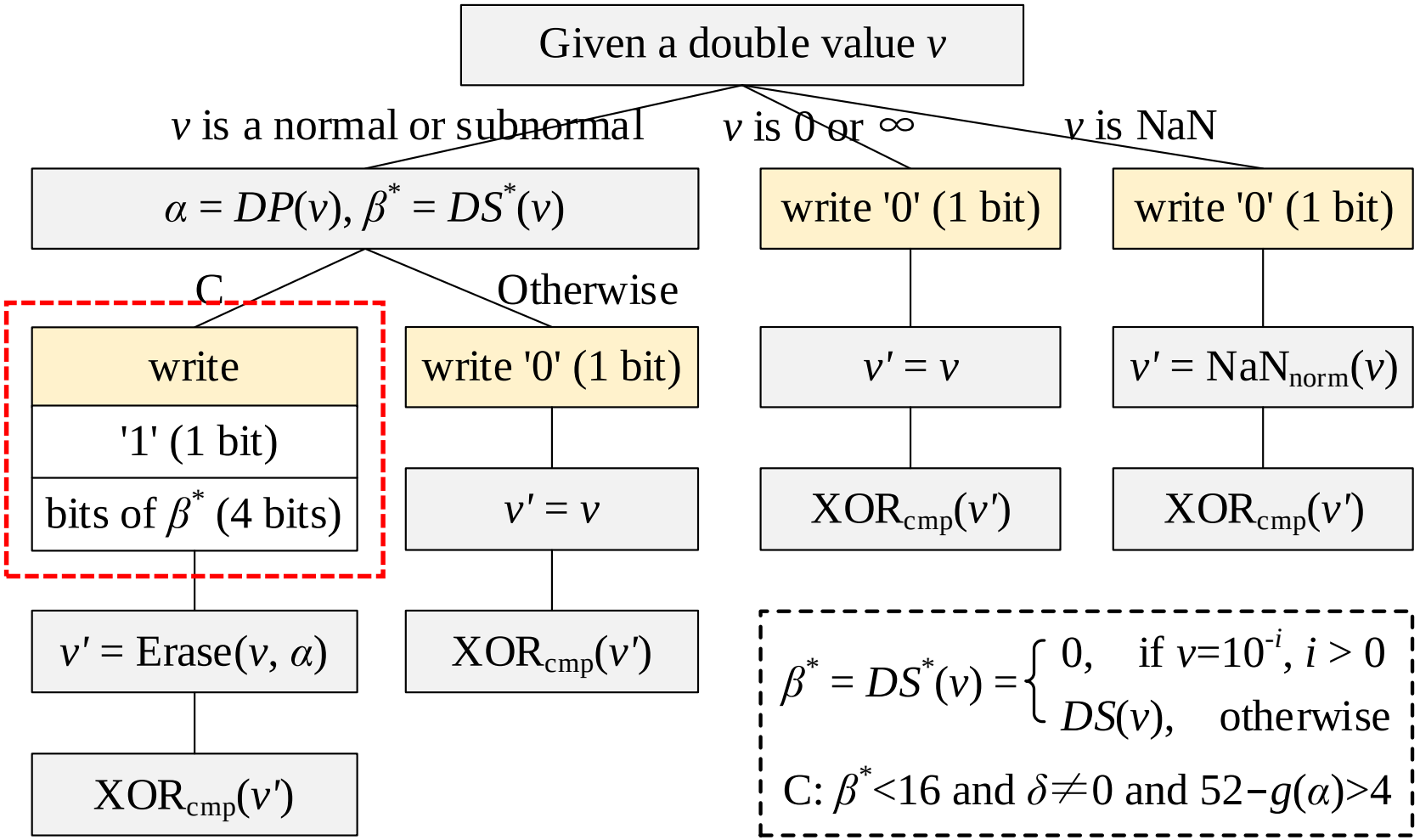
Below are comments from the review process:
The findings and experimental procedures presented in the study can be reproduced and validated using the methods described above and by utilizing the provided datasets and algorithms.

Among the three metrics, the compression ratio matched the paper perfectly, while the trends in compression time and decompression time were consistent with those described in the paper.

Ruiyuan Li, Zheng Li, Yi Wu, Chao Chen, Yu Zheng. Elf: Erasing-based Lossless Floating-Point Compression. (VLDB 2023, Accepted)

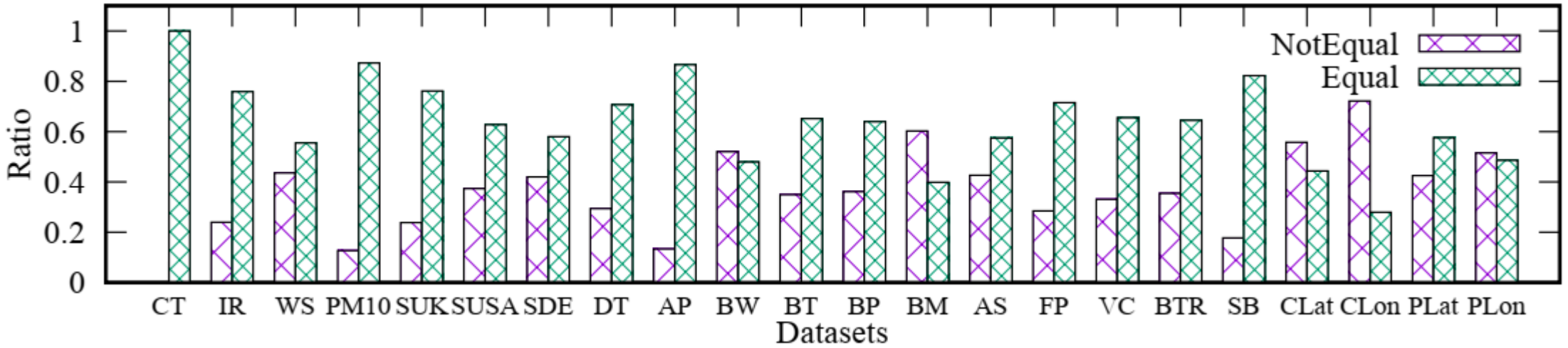
Optimization for Encoding β^* (Elf+)

- Elf writes a β^* (4 bits) for almost every value

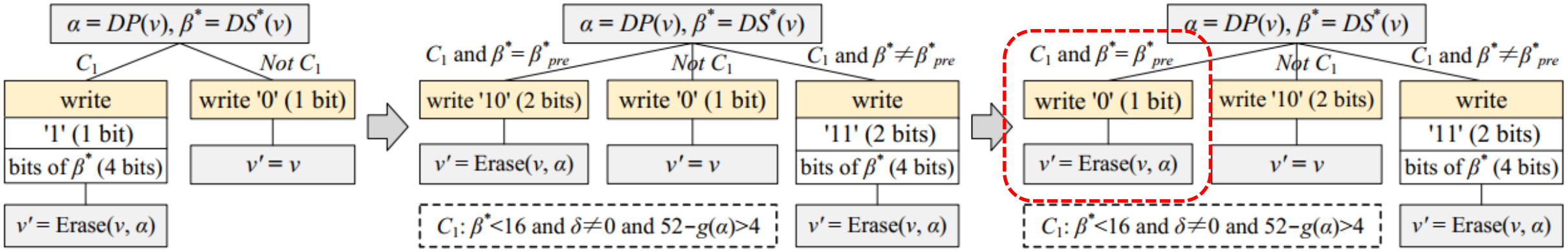


Optimization for Encoding β^* (Elf+)

- Values in the same time series tend to have the same significand count β



- Make use of β^*_{pre}. In most cases it takes only 1 flag bit



(a) Eraser in Algorithm 1

(b) Making Use of β^*_{pre}

(c) Reassigning Flag Code

Optimization for β Calculation (Elf+)

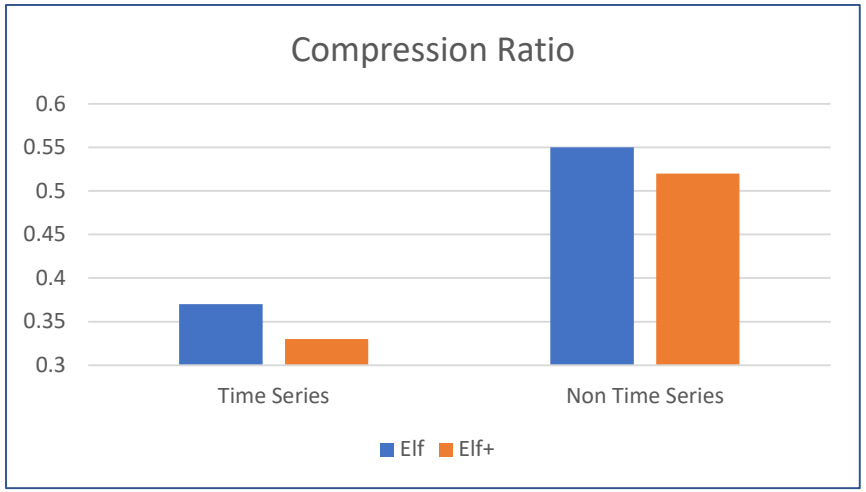
- **Basic Method**
  - Converted into String. E.g., “3.17”
  - Java BigDecimal

- **Elf Solution**
  - Enumerate i from $SP(v)$ to $SP(v) + 17$, until $v \times 10^i = \lfloor v \times 10^i \rfloor$
  - $SP(v) = \lfloor \log10|v| \rfloor$, $\beta = \mathrm{SP}(v) + i + 1$
  - E.g., $v = 3.17$, $SP(v) = 0$, $i^* = 2$, $\beta = 0 + 2 + 1 = 3$

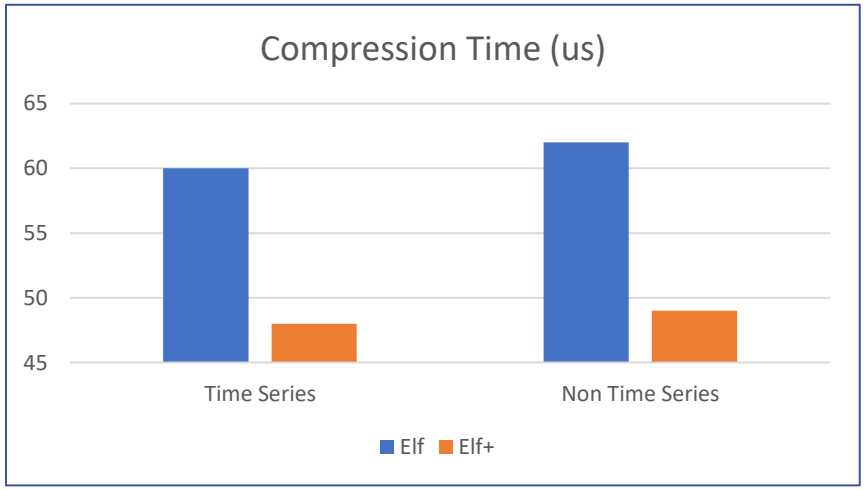
$O(17)$

- **Elf+ Solution**
  - Since $\beta_t = \beta_{t-1}$ in most cases, Elf+ starts i at $\beta_{pre} - SP(v) - 1$

$O(1)$



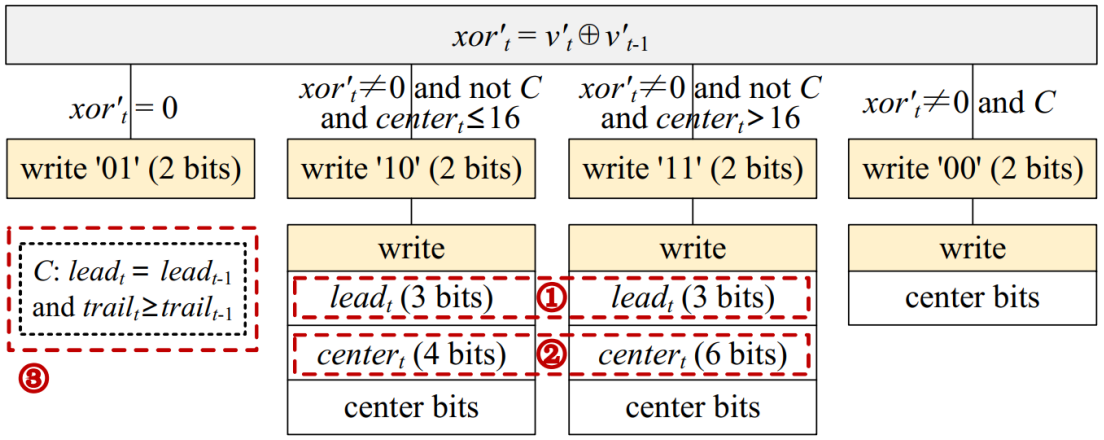
7.6%

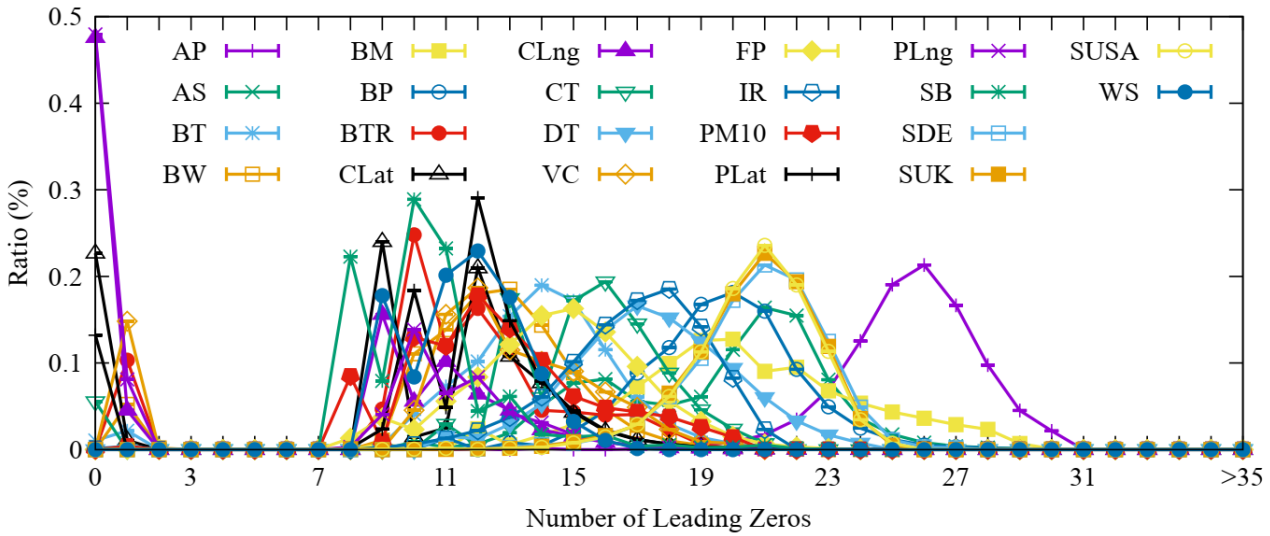


20.5%

Ruiyuan Li, et al. Erasing-based lossless compression method for streaming floating-point time series. (VLDBJ, Under Review)

Optimization for Encoding XORed Values (Elf*)





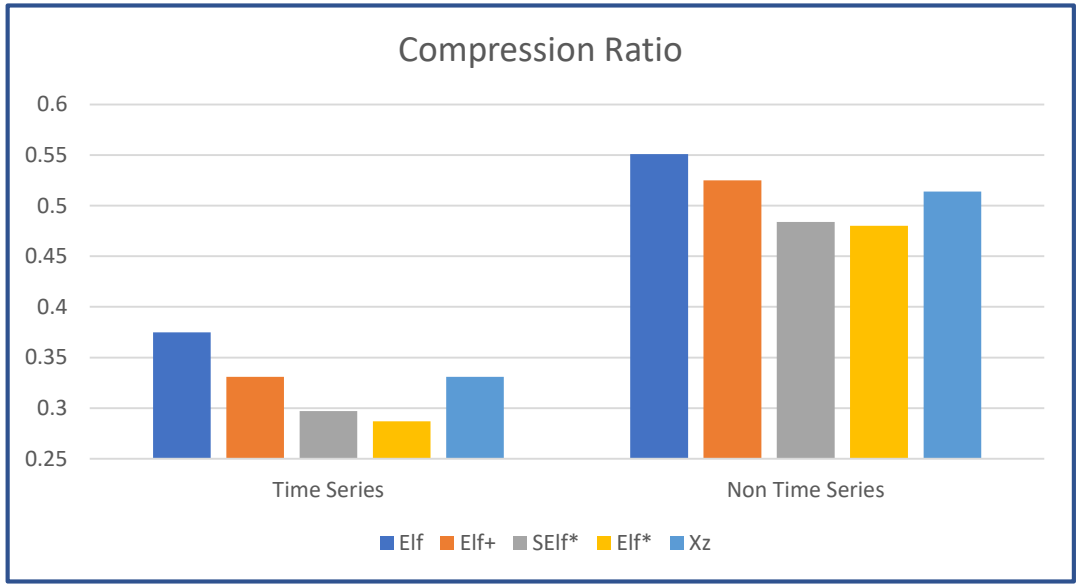
❑ Shortages of Elf's Encoding Strategies

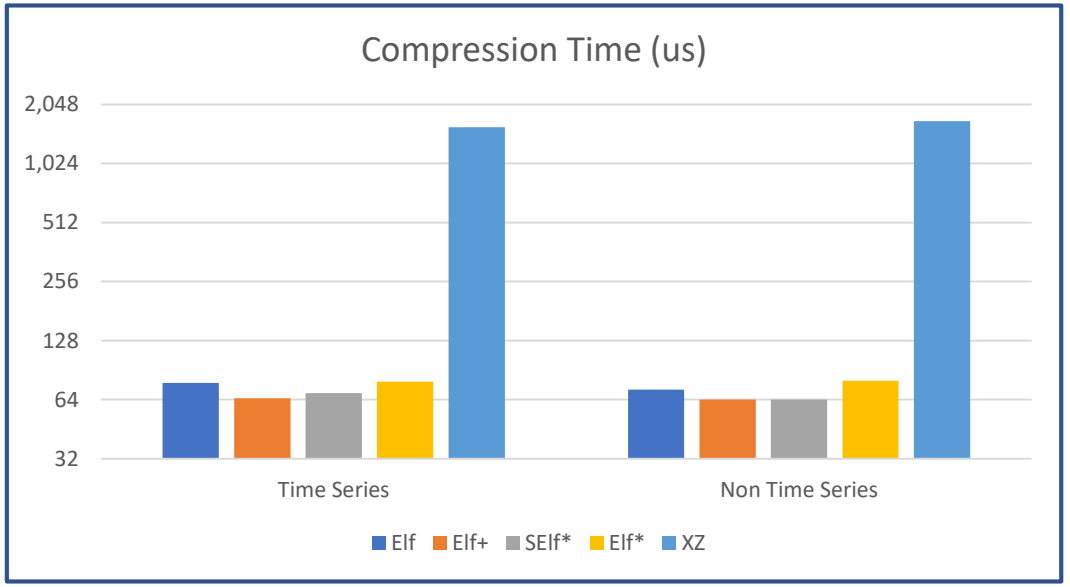
- ❑ Fixed Approximation Rule for *#lead*
  - ❑ $rule_{Elf}$ = (0, 8, 12, 16, 18, 20, 22, 24)
- ❑ 5/7 bits for *#center*
  - ❑ Including one flag bit
- ❑ Suboptimal Sharing Condition
  - ❑ Saved bits my not offset approximation cost

❑ Improvement by Elf*

- ❑ Adaptive Approximation Rule for *#lead*
  - ❑ Dynamic Programing with many pruning strategies
- ❑ Adaptively Encode *#trail* instead of *#center*
  - ❑ *#center* = 64 - *#lead* - *#trail*
- ❑ Adaptive Sharing Condition
  - ❑ Ensure a positive gain

Optimization for Encoding XORed Values (Elf*)





- **Streaming Elf* vs Elf+**
  - Compression Ratio ↑ 9.2%; Similar Compression Time
- **Elf* vs Xz**
  - Compression Ratio ↑ 10.1%; Takes Only 4.8% Compression Time

Ruiyuan Li, Zheng Li, Yi Wu, Chao Chen, Tong Liu, Yu Zheng. Adaptive Encoding Strategies for Erasing-Based Lossless Floating-Point Compression. (ICDE 2024, Under Review)

Start

Thank You!

重庆大学时空实验室
Chongqing University Start Lab

关注公众号，回复“Elf”，
下载海报、论文、PPT、源码等资料

时空实验室公众号