FAST'22

TVStore: Automatically Bounding Time-Series Storage via Time-Varying Compression

Yanzhe An, Yue Su, Yuqing Zhu✉, Jianmin Wang

E-mail: zhuyuqing@tsinghua.edu.cn

Tsinghua University 清華大學

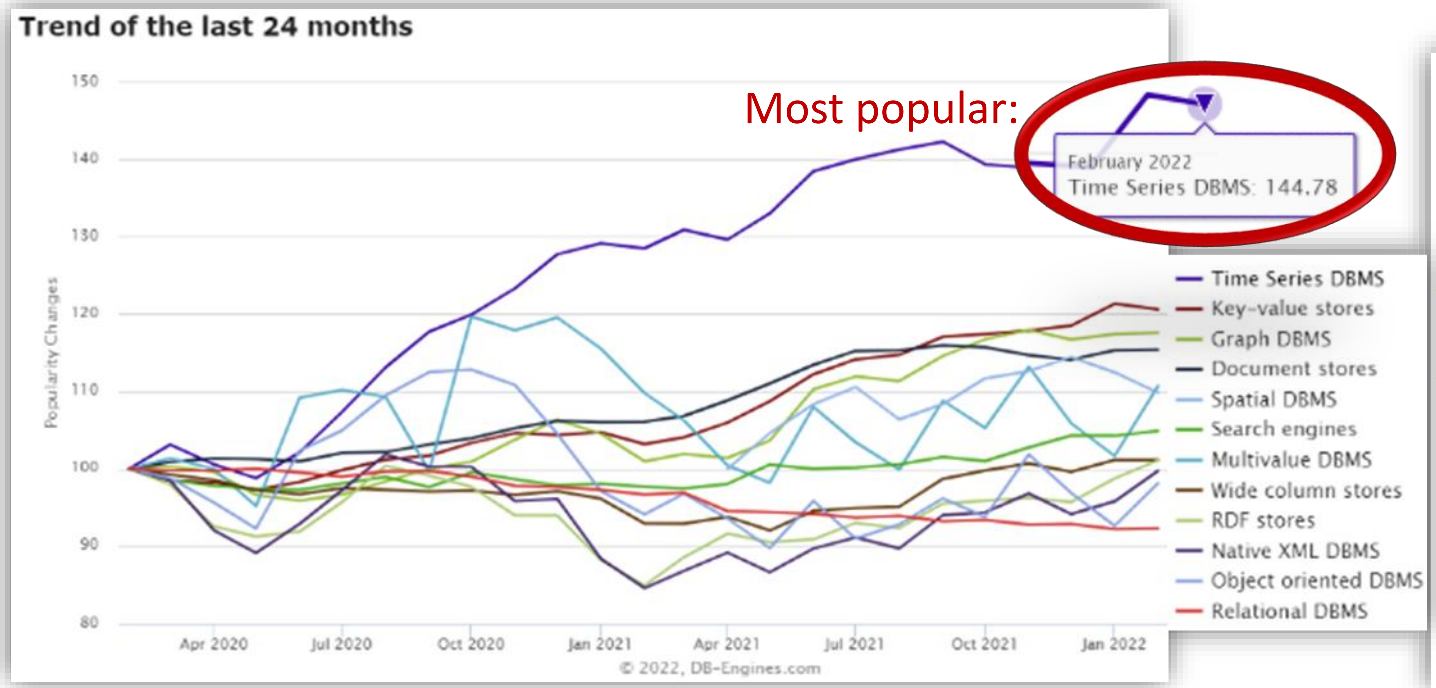
北京信息科学与技术国家研究中心
BEIJING NATIONAL RESEARCH CENTER FOR INFORMATION SCIENCE AND TECHNOLOGY

大数据系统软件国家工程实验室
National Engineering Laboratory for Big Data Software
NELBDS

Time Series Management: Popularity & Volume

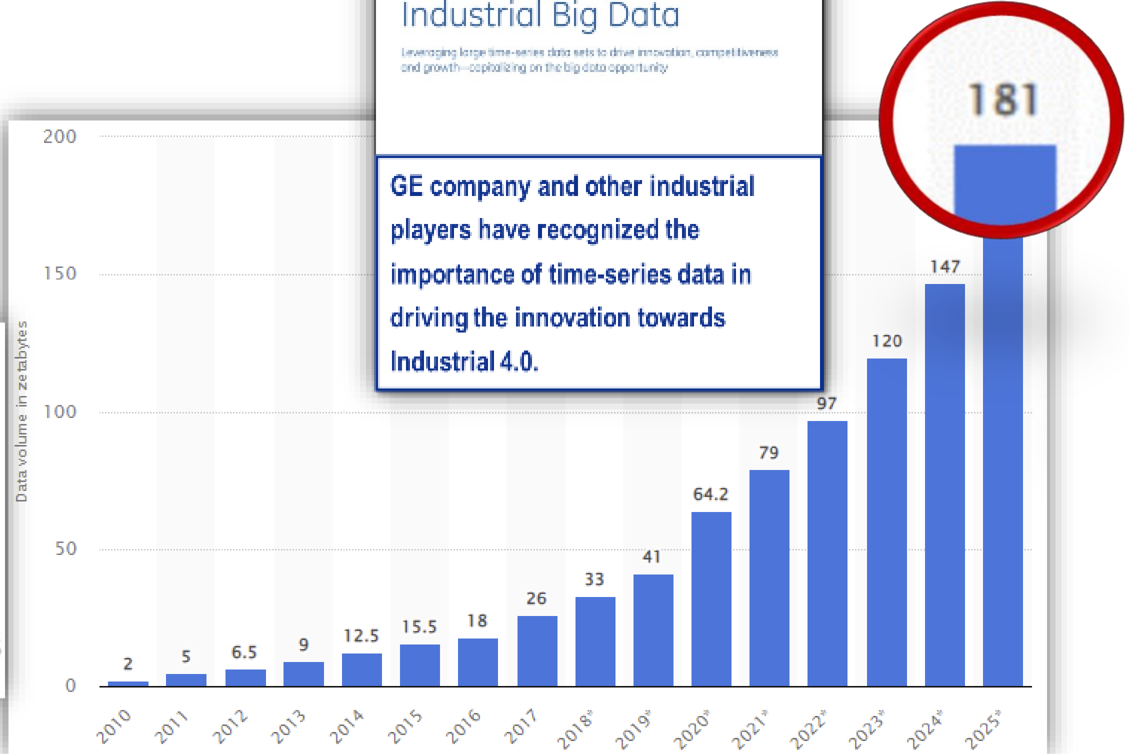
- Increasing popularity of time series management from wide adoption of:
  - Internet-of-Things devices, sensors; DevOps; Industrial 4.0
- Up-surging volume of data/information worldwide
  - ➔ overwhelming volume of time series data

Most popular:



GE company and other industrial players have recognized the importance of time-series data in driving the innovation towards Industrial 4.0.

Zetabytes



Motivation 1: Limited Resources and Expenses

- **Limited resources for applications:**
  - Limited satellite transmission bandwidth: 1TB/d for each oil platform at far sites
  - Unprecedentedly paramount data from scientific data: cosmology or meteorology
- **Limited expenses for users:**
  - Constrained expenses for increasing data volume: medium or small entities
  - Increasing costs due to increasing data volume: autonomous vehicles

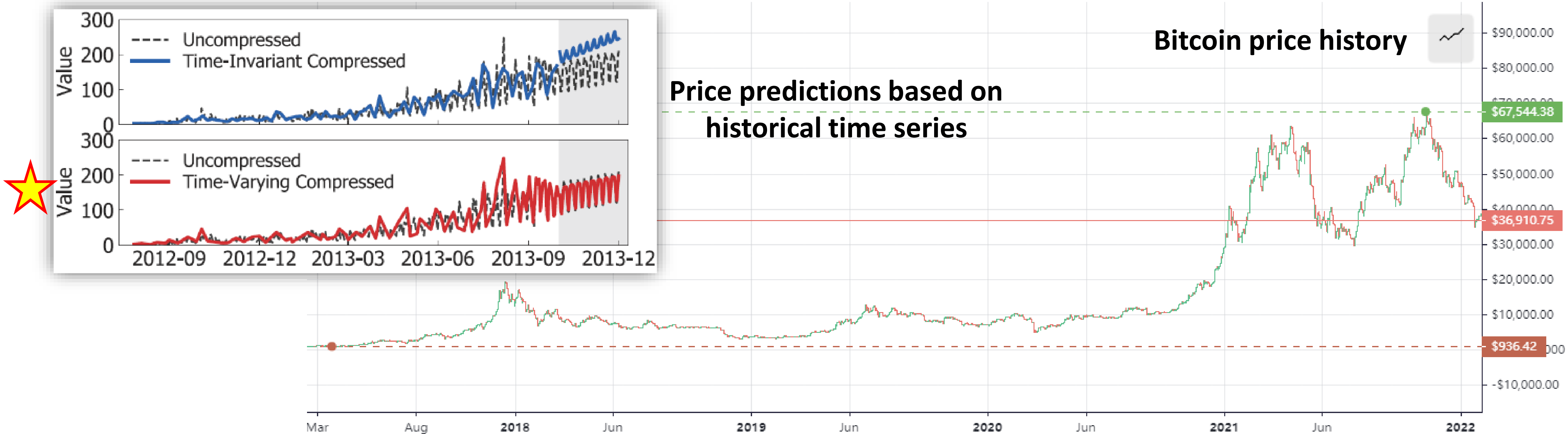
Cost saving by data compression at a high ratio

TSDB	Size (800GB)	Cost(USD)		Time of Range-Count Query: Secs (Error)		
		HDD	SSD	100%	80%	2%
InfluxDB(4X)	200	$10	$118	1347(0)	1263(0)	27(0)
TVStore(100X)	**8**	**$0.4**	**$4.8**	**1.8(0)**	**1.7(0.005)**	**0.7(0)**

Motivation 2: Time-Varying Importance of Data

- **The importance of time series data changes along with time.**
  - As reflected by applications' favoring recent data over old data, or favoring some events at certain moments over others
  - A feature commonly existing in social, natural, and scientific phenomena
  - E.g., price predictions for stock or cryptocurrency market

Faster and more accurate predictions on the same small volume of data



Our approach and related work

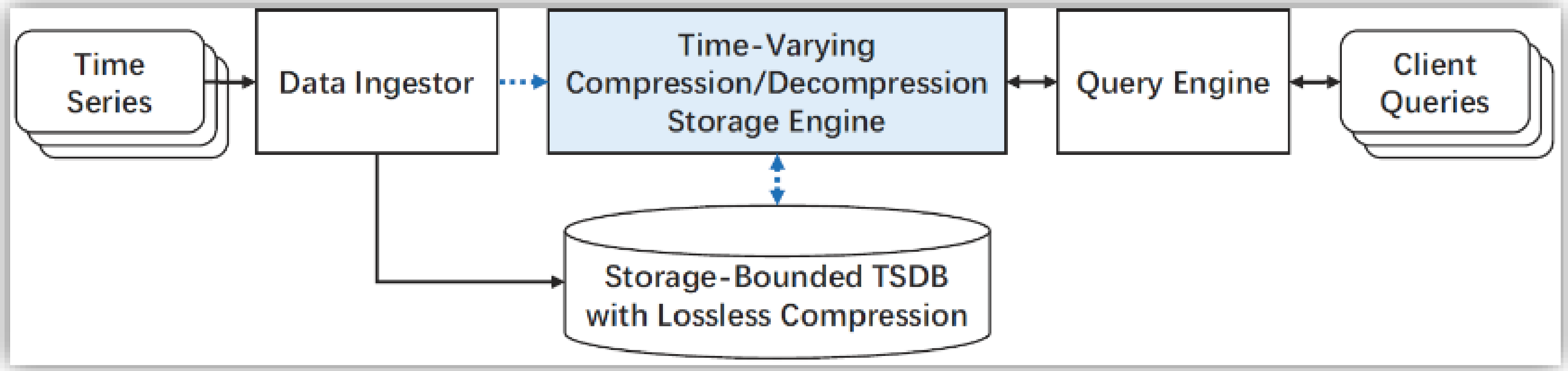
- **Our key insight and major approach:**
  - Time-varying compression compresses data complying with the importance of data.
  - Automatically bounding storage by time-varying compression to reduce costs
    - Run the time-varying compression framework at proper times

- TSDB (time series database) with time-invariant compression
  - Lossless compression: limited compression ratio and volume reduction
  - Lossy compression: fixed trade-off between storage and accuracy
- TSDB with bounded storage: losing all information on deleted data
  - By retention policy with time-based deletion: InfluxDB
  - By storage recycling in the round-robin way: RRDTool
- Recent work: SummaryStore keeps predefined time-decaying summaries, without bounding storage.

TVStore Overview

- Featured by time-varying de-/compression storage engine
  - Other components remain consistent with the host TSDB

➔originally supported database functions can still be supported



① Compressing data in a time-varying manner

- By user-defined compression ratio and time-dependent function

② Bounding storage automatically

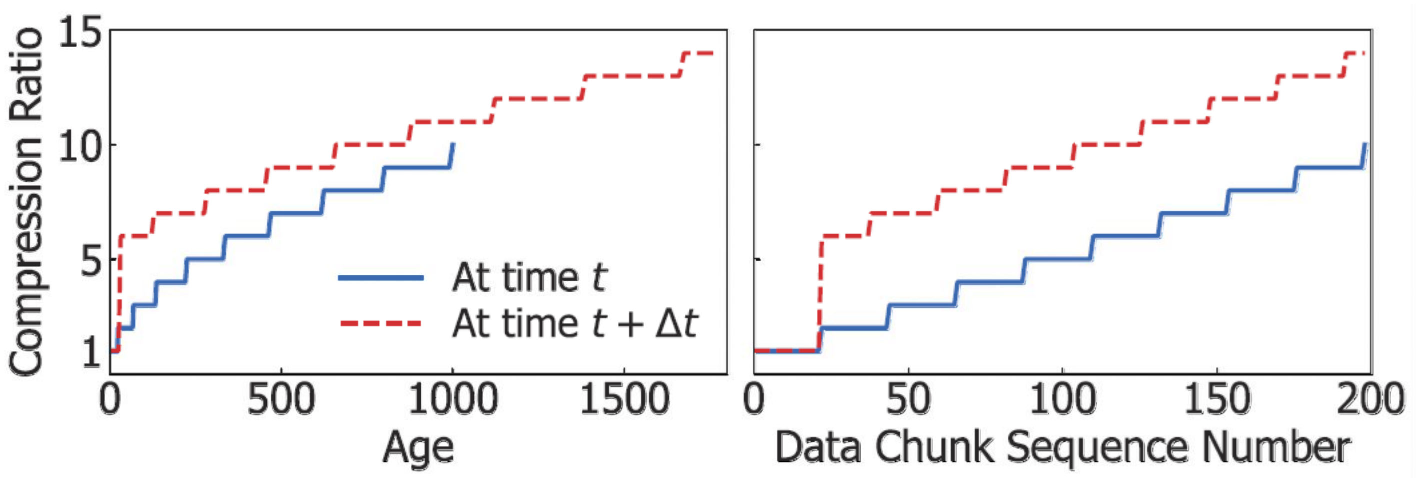
- To the user-specified storage size

Time-Varying Compression

- **Key question**
  - How to compress according to a time-varying function efficiently, as data keep being ingested?

    ➔Each piece of data must be compressed to different ratios at different times.

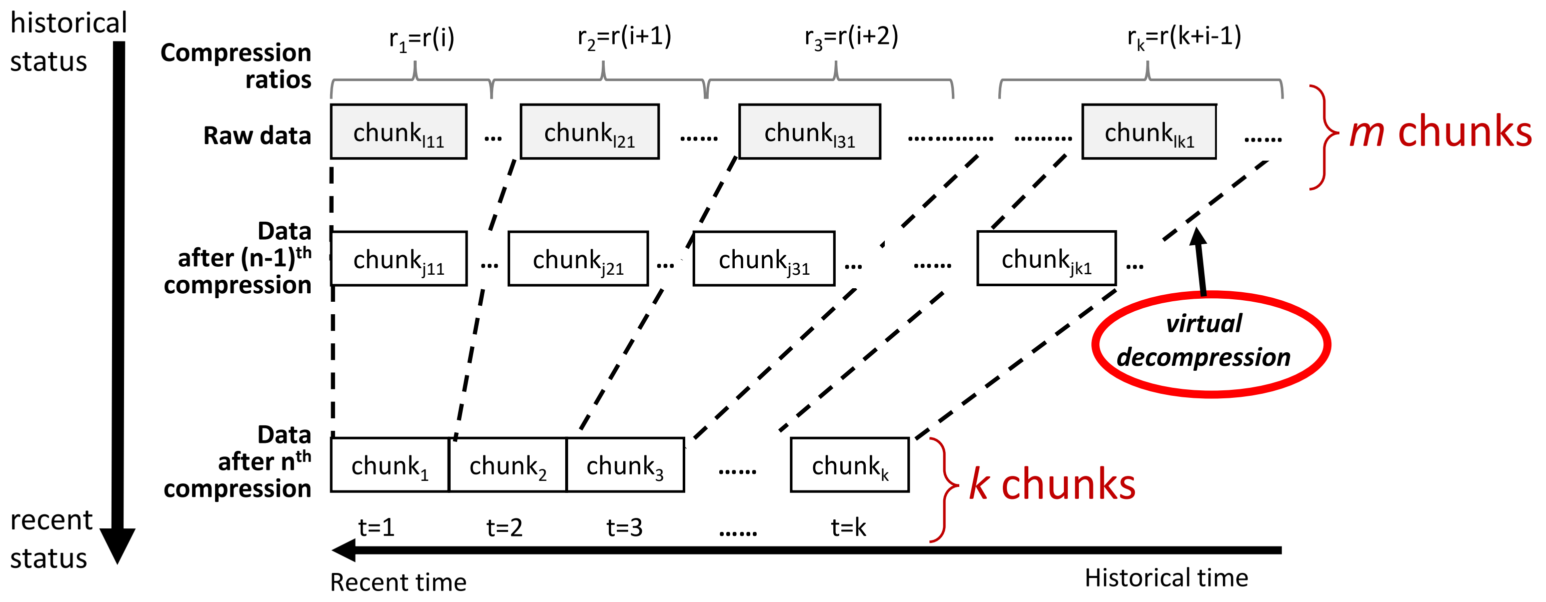
    ➔Compression and decompression take time.



Time-Varying Compression

- **Key techniques**
  - **Virtual decompression**
    - Map to the raw data size for re-compression

    ➔Exempting the cost of decompression



  - **Ratio compliance by approximation**

$$\Sigma_i^k r_i \geq m \quad (1)$$

$$m/k \geq \bar{r} \quad (2)$$

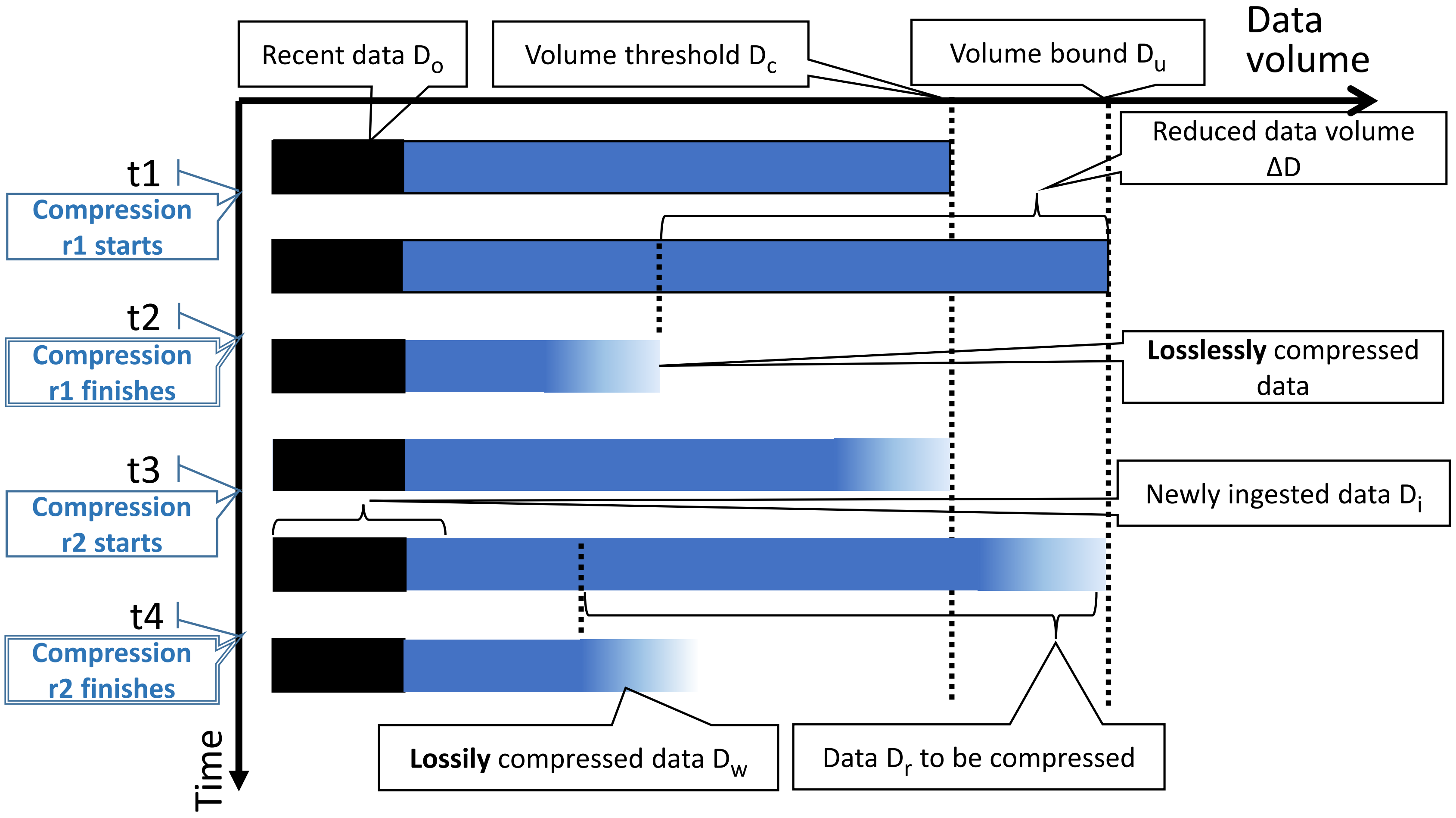
The number *m* of raw data chunks

The target compression ratio

The number *k* of target data chunks

Design choices for automatic bounding

- The automatic storage bounding process on fast data ingestion



Design choices for automatic bounding

- **Key questions**
  ① How to compress?
  - Compression on hot data or cold data

  ➔ **Fewer** compression rounds & computation costs

  ② What ratio to compress?
  - **Proper** compression ratio interval

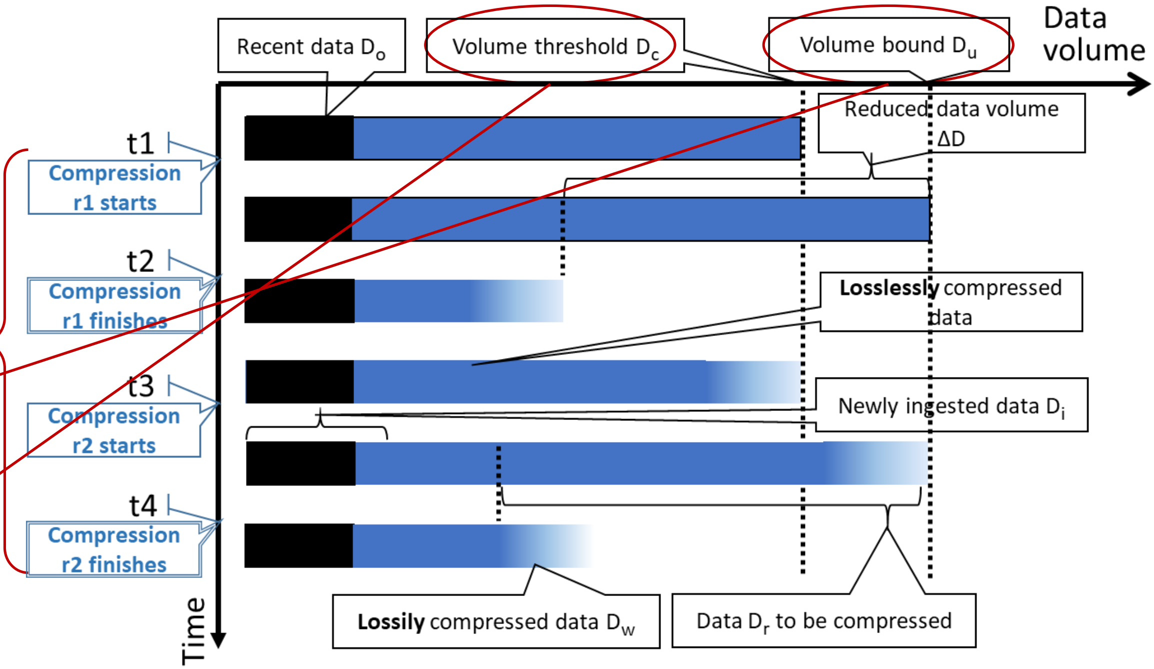
  ➔ Too large: losing information unnecessarily

  ➔ Too small: exceeding storage bound

  ③ When to compress?
  - **Proper** compression initiation time

  ➔ Too early: losing information unnecessarily and involving unnecessary costs

  ➔ Too late: exceeding storage bound



Design choices for automatic bounding

- Theoretical deductions on **the decision and tight bounds:**

① **How to compress?** Cold data compression is better.

> **Principle 1.** *For a given range of time series data and a sequence of compression ratios, iterative compressions over cold data can reduce the compression rounds as compared to the continuous compression method on hot data.*

② **What ratio to compress?** $\boxed{r_c \geq \frac{v_r}{v_r - v_i}}$

> **Principle 2.** *To avoid overrunning a storage bound, the compression ratio r_c for each round of compression must be no smaller than $\frac{v_r}{v_r - v_i}$, where v_r is the average read throughput from the disk and v_i is the ingestion throughput by applications.*

③ **When to compress?** $\boxed{D_c \leq (D_u - D_o)/(\frac{v_i}{v_r} + \frac{1}{r} + 1) + D_o}$

> **Principle 3.** *Let D_u be the bound on the storage space and D_o be the recent data not to be compressed. Let v_r be the average read throughput from the disk and v_i the ingestion throughput by applications. Given the compression ratio $\bar{r}$ for a compression round, the threshold D_c of data volume to start a compression must satisfy the following condition.*
>
> $$D_c \leq (D_u - D_o)/(\frac{v_i}{v_r} + \frac{1}{r} + 1) + D_o \qquad (15)$$

Experimental Settings

- Datasets

Real-world datasets	REDD public dataset **(7.5TB)**		Train-load private dataset **(6.6TB)**
Synthetic datasets	Uniform random **(5TB)**	Poisson distribution **(5TB)**	Pareto distribution **(5TB)**

- Workloads

Ingestion	With compression ratios at 1X, 20X, 60X, 100X
Query	Aggregations (sum, avg, max, min) for data at Age(*S*) with Length(*S*), *S*=(Mon/Millennia, Day/Century, Min/Recent)

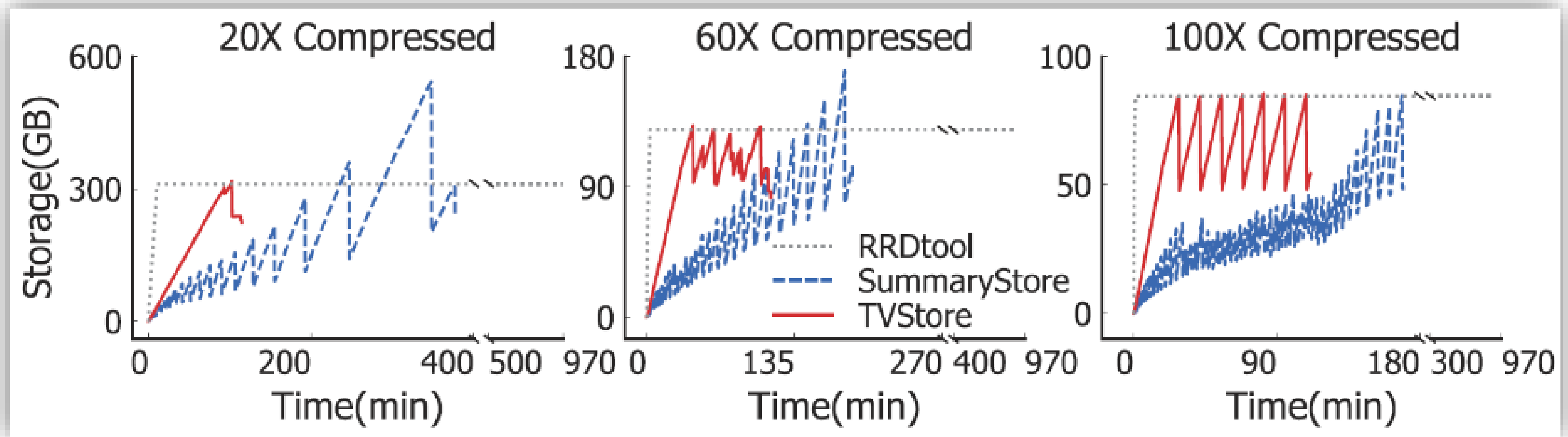
- Compared systems

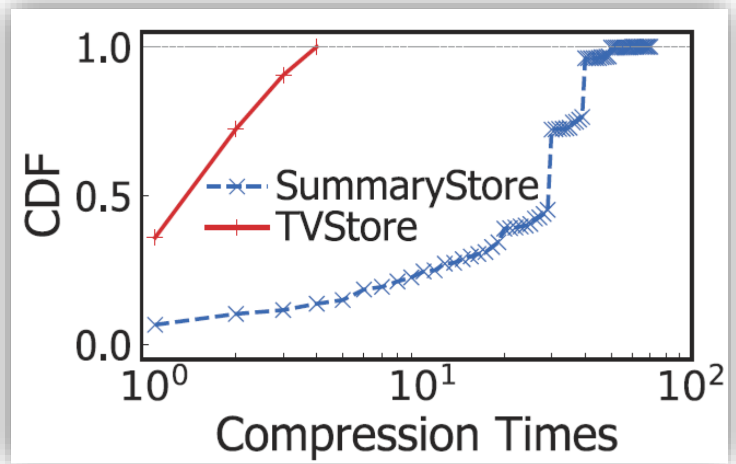
SummaryStore	**RRDTool**	**Apache IoTDB**
Approximate time series store	Round-robin time series DB	Implementation baseline

- Hardware instances:
  - Setting 1: two Intel Xeon E5-2650 CPUs, 370GB DDR4 memory
  - Setting 2: 32GB memory and an 8-core CPU

Storage bounding & compression cost

- **TVStore effectively bounds its storage with high ingestion performance.**
  - RRDTool bounds storage with low ingestion performance.
  - SummaryStore does not support storage bounding.



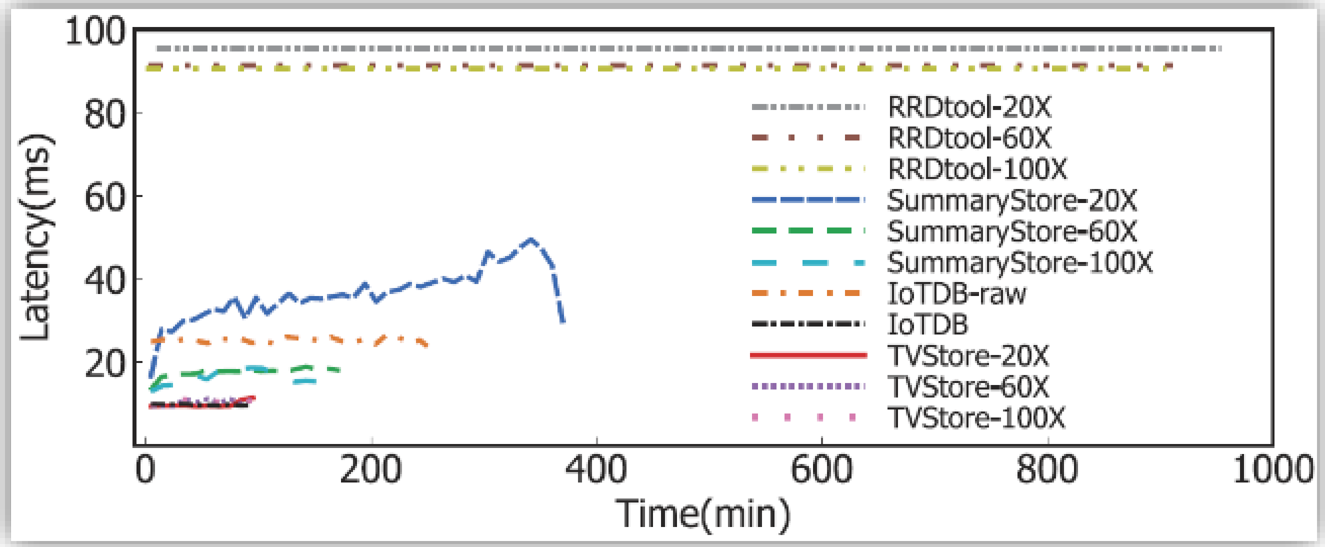


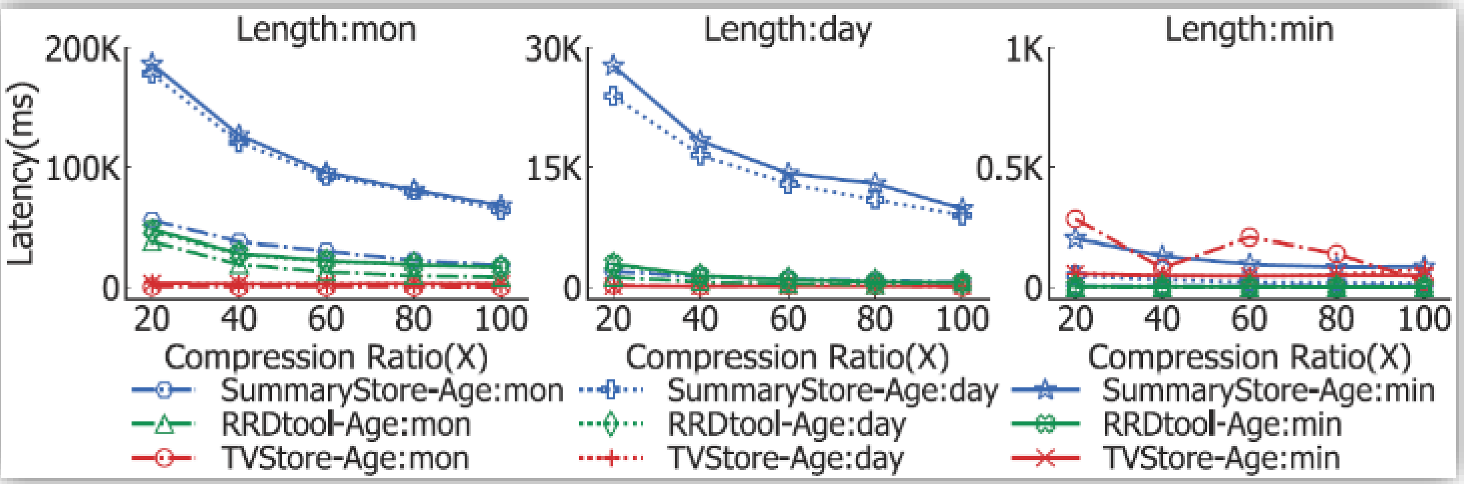
- **TVSTore requires fewer compression/merging times than SummaryStore.**
  - Incurring fewer disk I/Os and computation costs
  - Cold-data compression is more efficient than hot-data compression.

Ingestion & query performances

- TVStore has much **higher ingestion throughput** than SummaryStore and RRDtool in all cases.

- TVStore's compression process has **little impact** on the normal processing of writes.



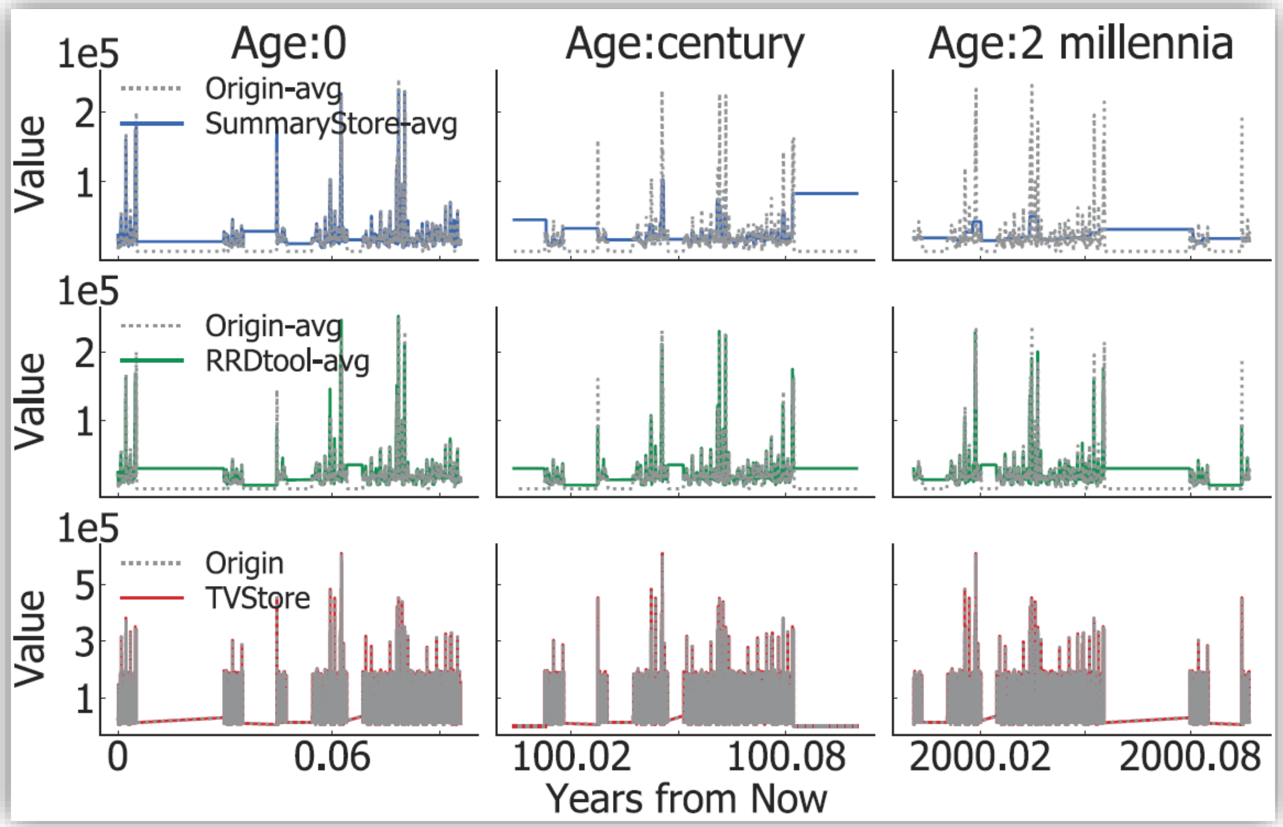


- TVStore implementation can answer queries **35X and 8.7X faster** than SummaryStore and RRDtool respectively for the best case.

How data look in databases

- Time-varying pattern
  - TVStore and SummaryStore demonstrate **time-varying patterns**, while RRDtool has the time-invariant curves.

- **Preserving much more information**
  - Under the same overall data reduction/compression ratio, TVStore can restore data to almost **the same as the original**, while RRDtool and SummaryStore cannot.



Takeaways and future work

- **Storage bounding is possible in ways other than directly discarding data.**
  - TVStore bounds storage **GRADUALLY** and **AUTOMATICALLY.**
- **Data can be compressed according to a time-varying function.**
  - TVStore supports user-defined function in its **time-varying compression framework**.
- **Future work**
  - TVStore supports plug-in time-varying functions.

    ➔ How to decide the best function for an application
  - TVStore supports plug-in compressors.

    ➔ How to decide the best compressor for an application

    ➔ **Using learned models as lossy compressors**

TVStore: Automatically Bounding Time-Series Storage via Time-Varying Compression

Open-source: https://github.com/thulab/TVStore

Thank you!

Yanzhe An, Yue Su, Yuqing Zhu✉, Jianmin Wang

E-mail: **zhuyuqing@tsinghua.edu.cn**