CS 839: Design the Next-Generation Database

Lecture 25: Time Series

Xiangyao Yu

4/21/2020

Announcements

Next lecture: Guest lecture by Dr. Shasank Chavan from Oracle

Final presentation schedule:

Day 1 (Tuesday, April 28)	**Project Name**
1:05 -- 1:15	Scheduling for HTAP systems on CPU-GPU clusters
1:15 -- 1:25	Pushing Databases to the Edge
1:25 -- 1:35	Graph Embedding on Diverse Deployment Architectures: An Experimental Study
1:35 -- 1:45	Experimental Analysis of Graph Database Systems
1:45 -- 1:55	A survey of GPU accelerated database systems
1:55 -- 2:05	Machine learning in databases: Where are we now, where can we go?
2:05 -- 2:15	Performance comparison of Deferred Lock Enforcement and Control Lock Violation
Day 2 (Thursday, April 30)	**Project Name**
1:05 -- 1:15	Frontend and Query Scheduling Engine Based on SQLite3
1:15 -- 1:25	Exploiting Accelerator Level Parallelism to Accelerate Database Analytics
1:25 -- 1:35	The Poor Man's Ferrari: A New NVM-based Database Architecture
1:35 -- 1:45	RJoin: A join algorithm for RDMA networks
1:45 -- 1:55	Graph Databases: A Survey on Models and System Designs
1:55 -- 2:05	Opportunities for adaptive concurrency control protocols

Discussion Highlights

Challenges of applying VM-snapshot to a shared-memory OLTP system?

- The additional burden in taking a consistent snapshot.
- Consistent snapshot across machines is a hard distributed system problem.
- Fork() becomes more expensive

How to reduce the cost of fork() when database is large?

- Large pages and partitioning
- Copy on write on the page table itself
- Append child parameters to parents page table (one page table for all processes)
- Share the page tables with OLTP while recording changed pages with a side data structure.
- Fork() performs CoW at tuple granularity

Most promising architecture of HTAP? (single vs. separate systems, shared vs. separate data)

- Single system, separate data: since optimal data formats (row vs column) for OLTP and OLAP.
- Single system, shared data: different data formats (row/column) in different replicas; avoid data sync and update propagation
- Separate system and separate data: no interference between OLTP and OLAP.
- Separate system and shared data: better scalability and high availability

Today's Paper

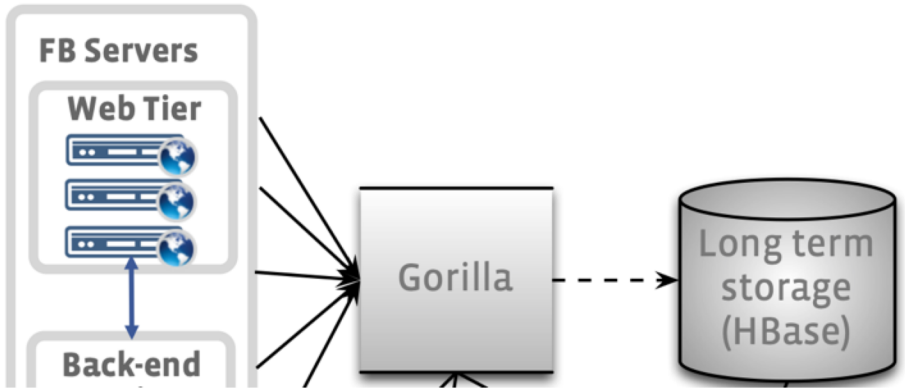
Gorilla: A Fast, Scalable, In-Memory Time Series Database

Tuomas Pelkonen Scott Franklin Justin Teller
Paul Cavallaro Qi Huang Justin Meza Kaushik Veeraraghavan
Facebook, Inc.
Menlo Park, CA

ABSTRACT

Large-scale internet services aim to remain highly available and responsive in the presence of unexpected failures. Providing this service often requires monitoring and analyzing tens of millions of measurements per second across a large number of systems, and one particularly effective solution is to store and query such measurements in a time series database (TSDB).

A key challenge in the design of TSDBs is how to strike



VLDB 2015

What is a Time Series Database (TSDB)?

According to Wikipedia

> A time series database (TSDB) is a software system that is optimized for storing and serving time series through associated pairs of time(s) and value(s)

What is a Time Series Database (TSDB)?

According to Wikipedia

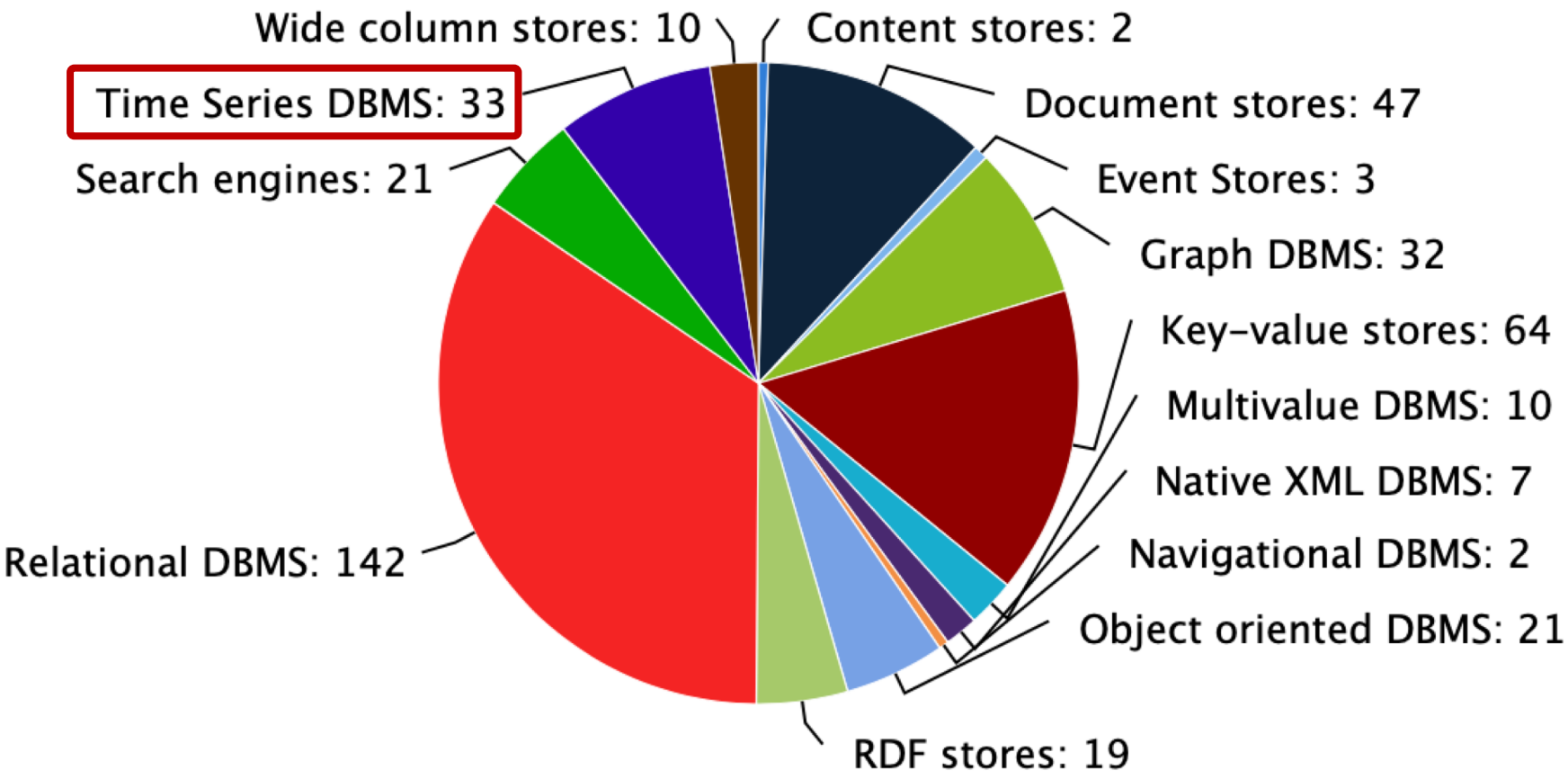
> A time series database (TSDB) is a software system that is optimized for storing and serving time series through associated pairs of time(s) and value(s)

Q: Can I use a normal database with a column called “timestamp”?

A: Yes, you can. But performance will suffer.

Database Popularity

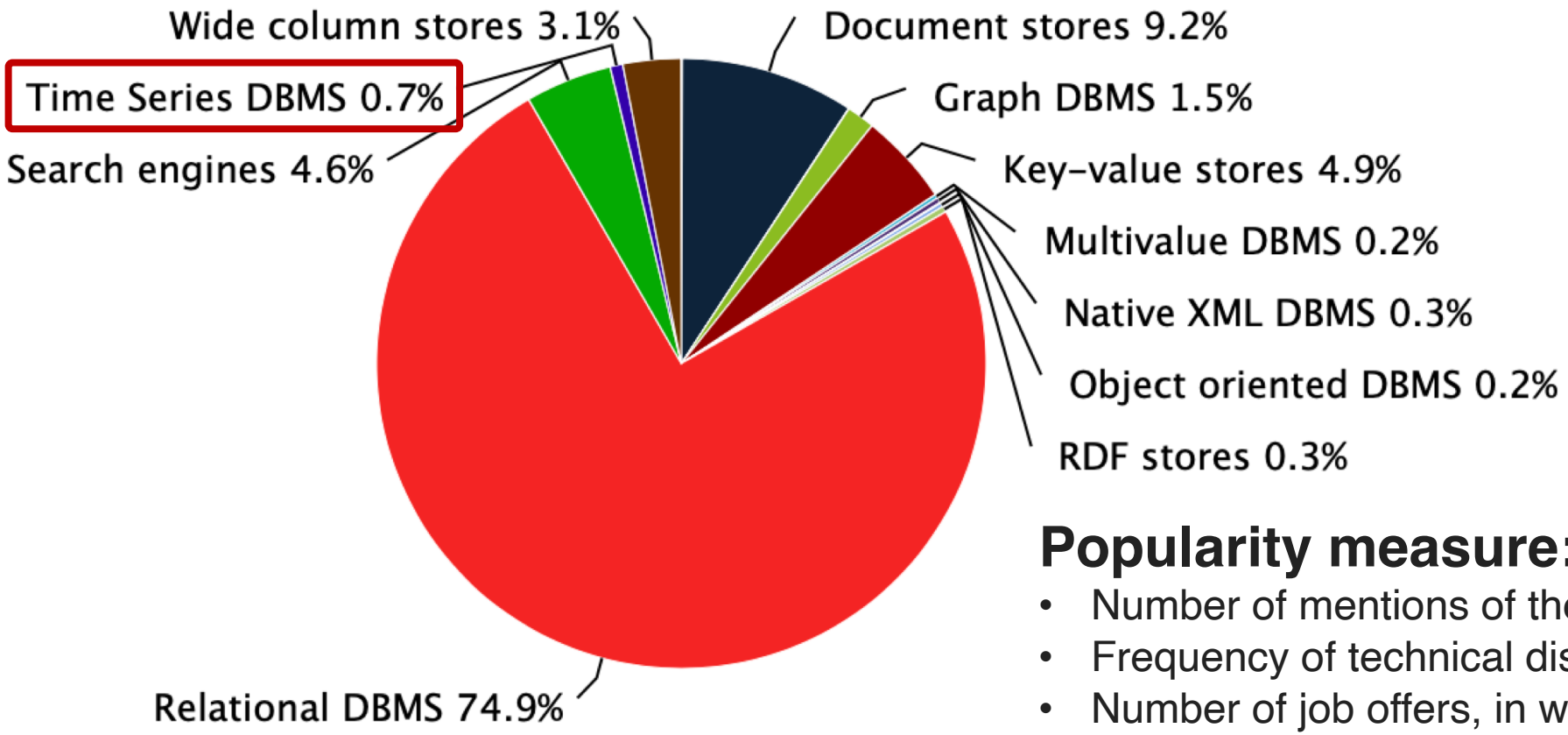
Number of systems per category (355 systems in total), April 2020



Source: DB-Engines https://db-engines.com/

Database Popularity

Ranking scores per category in percent, April 2020

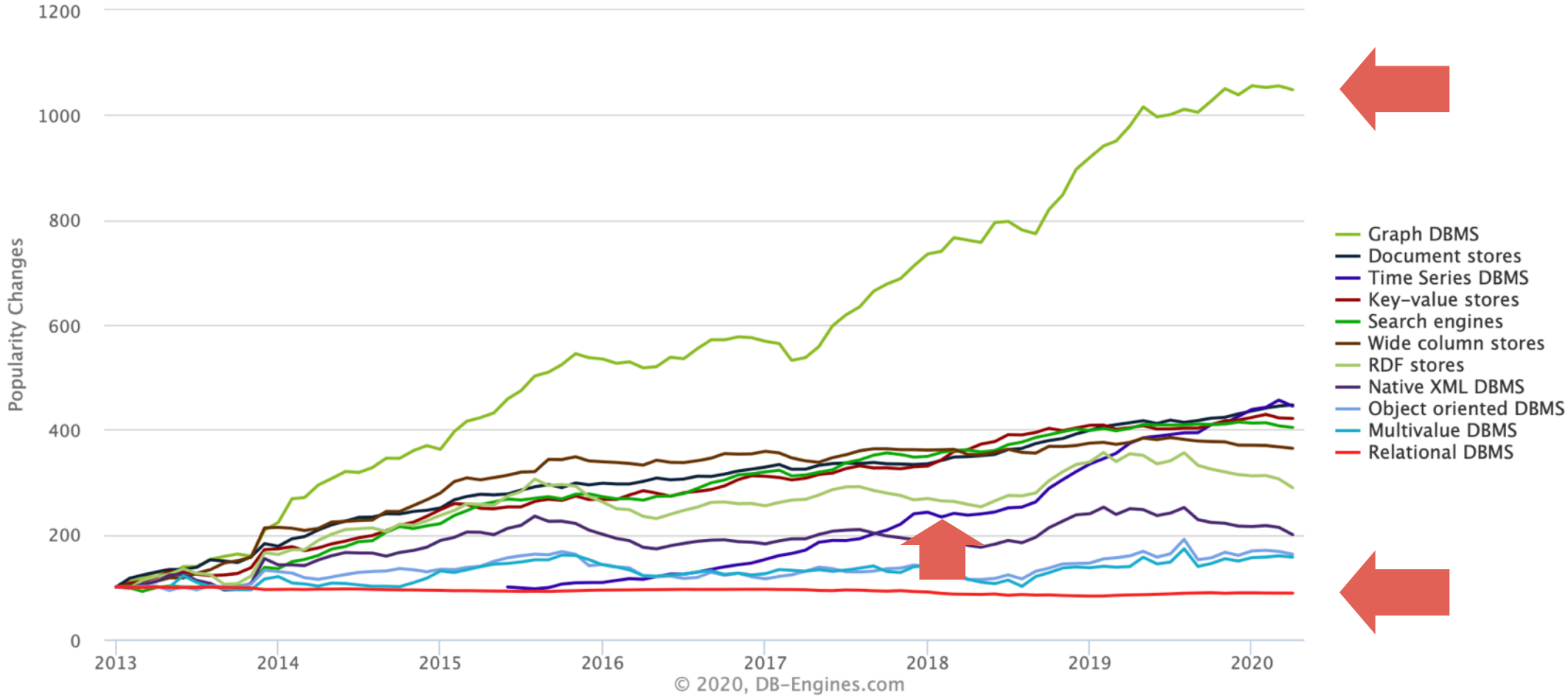


Source: DB-Engines https://db-engines.com/

Popularity measure:

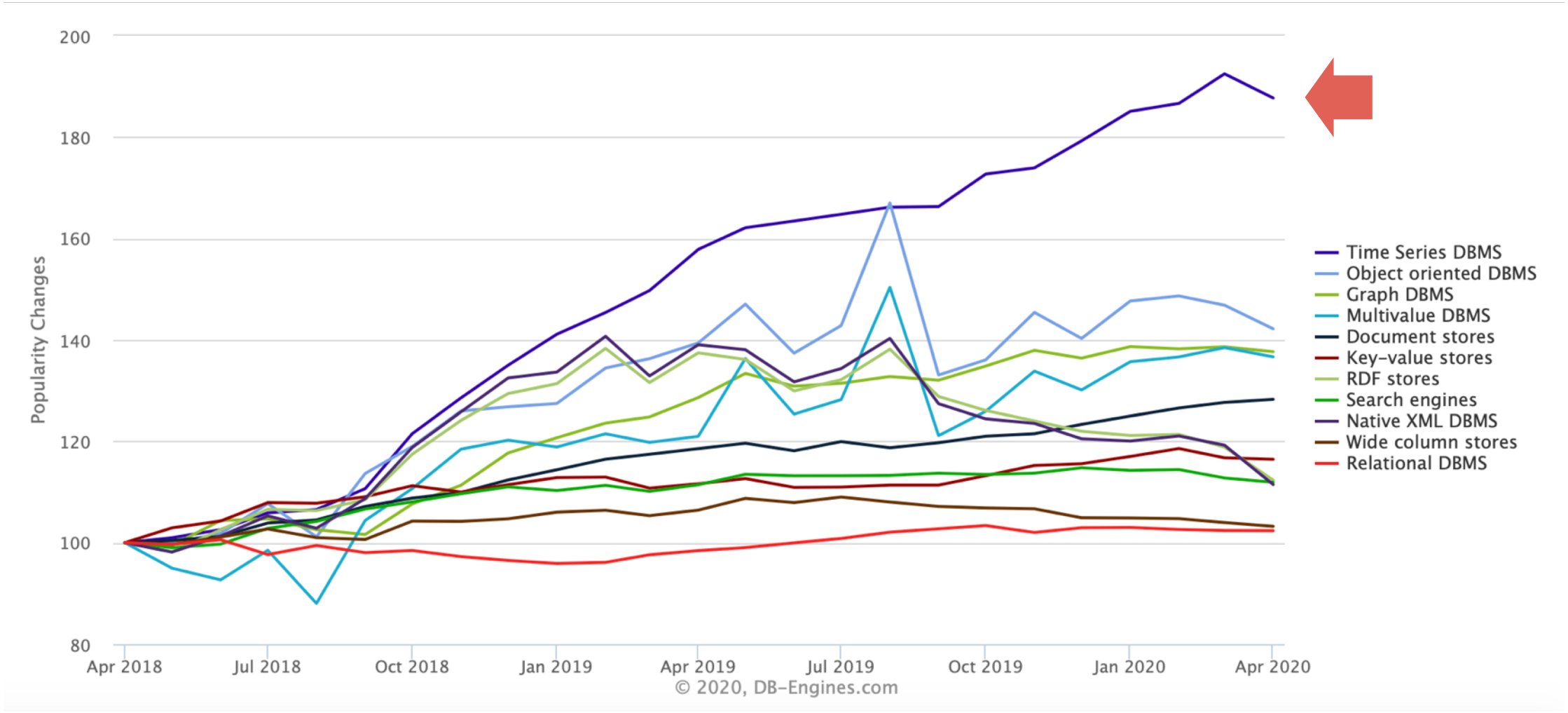
- Number of mentions of the system on websites
- Frequency of technical discussions about the system.
- Number of job offers, in which the system is
- Number of profiles in professional networks, in which the system is mentioned.
- Relevance in social networks.

Trend Since 2013



Source: DB-Engines https://db-engines.com/

Trend of the Last 24 Months



Source: DB-Engines https://db-engines.com/

Why is TSDB Popular?

Monitoring software systems: Virtual machines, containers, services, applications

Monitoring physical systems: Equipment, machinery, connected devices, the environment, our homes, our bodies (Internet of Things)

Asset tracking applications: Vehicles, trucks, physical containers, pallets

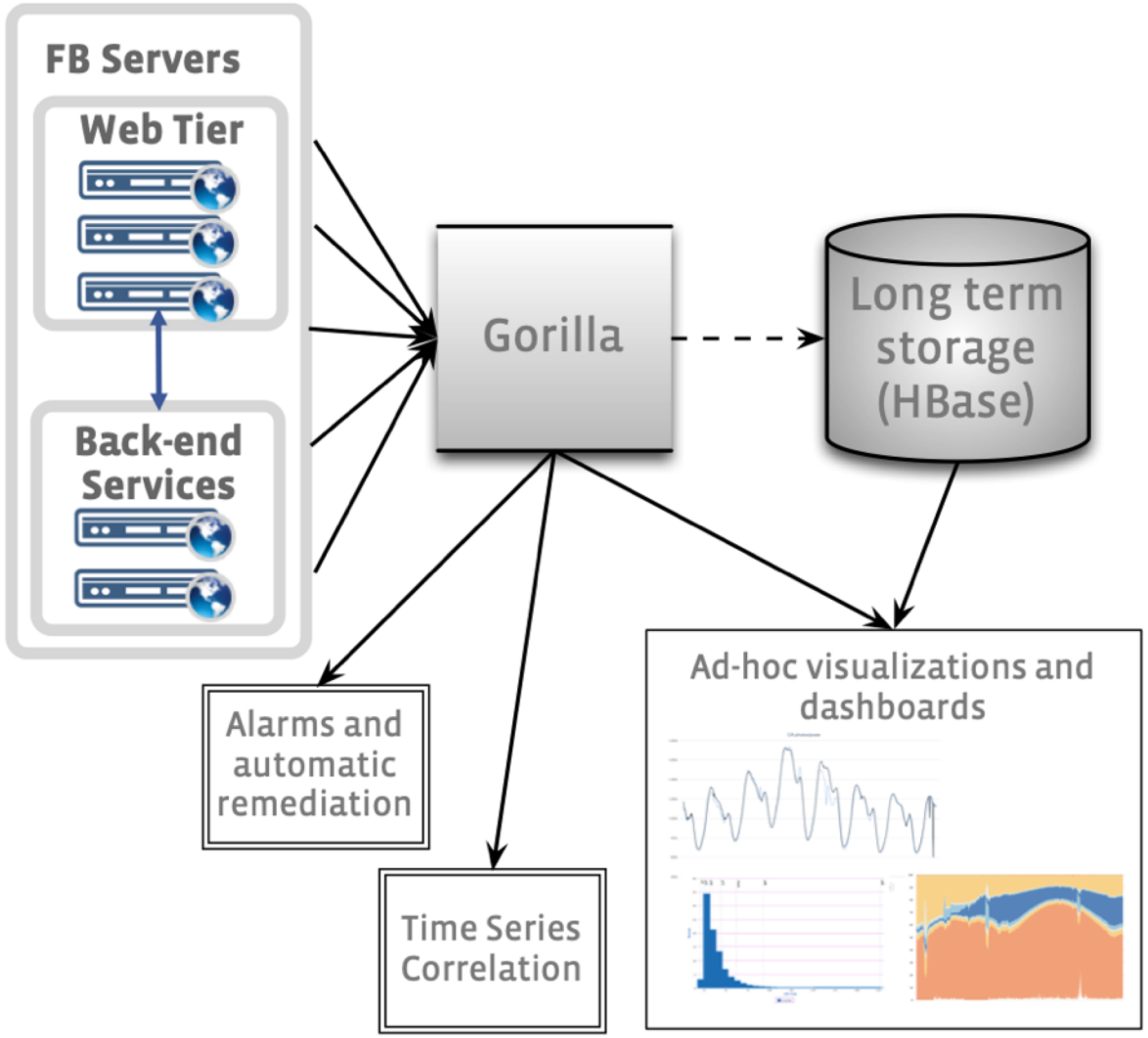
Financial trading systems: Classic securities, newer cryptocurrencies

Eventing applications: Tracking user/customer interaction data

Business intelligence tools: Tracking key metrics and the overall health of the business

Source: What the heck is time-series data (and why do I need a time-series database)?
https://blog.timescale.com/blog/what-the-heck-is-time-series-data-and-why-do-i-need-a-time-series-database-dcf3b1b18563/

Gorilla



An in-memory cache for the slower TSDB on HBase

Time Series Compression

Data format: (timestamp, value)

- Timestamp: 64-bit
- Value: 64-bit double-precision floating point value

02:00:01	10.2
02:00:02	11.3
02:00:03	11.8
02:00:04	12.2

...

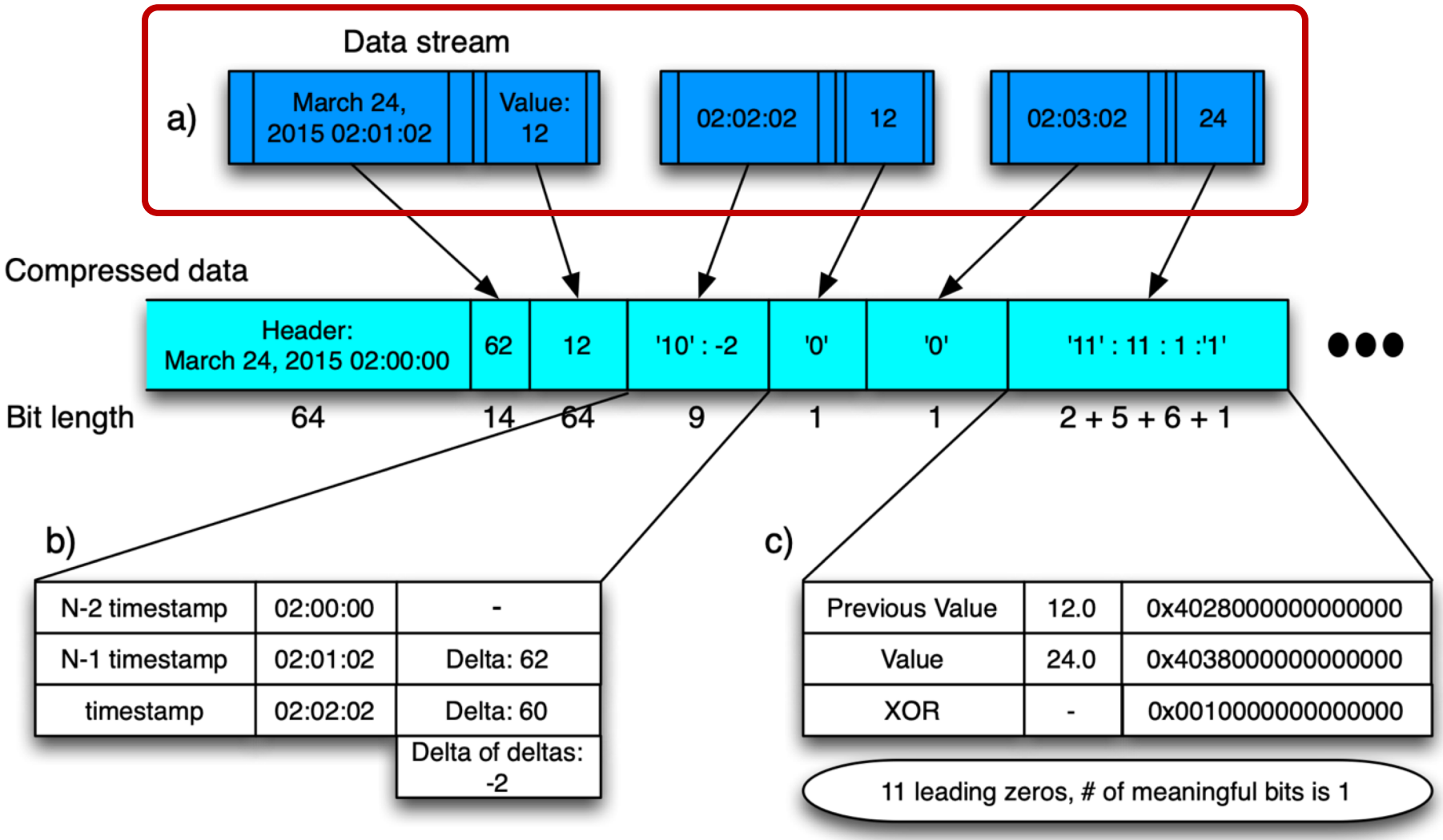
Timestamp compression

- Delta-of-delta

Value compression

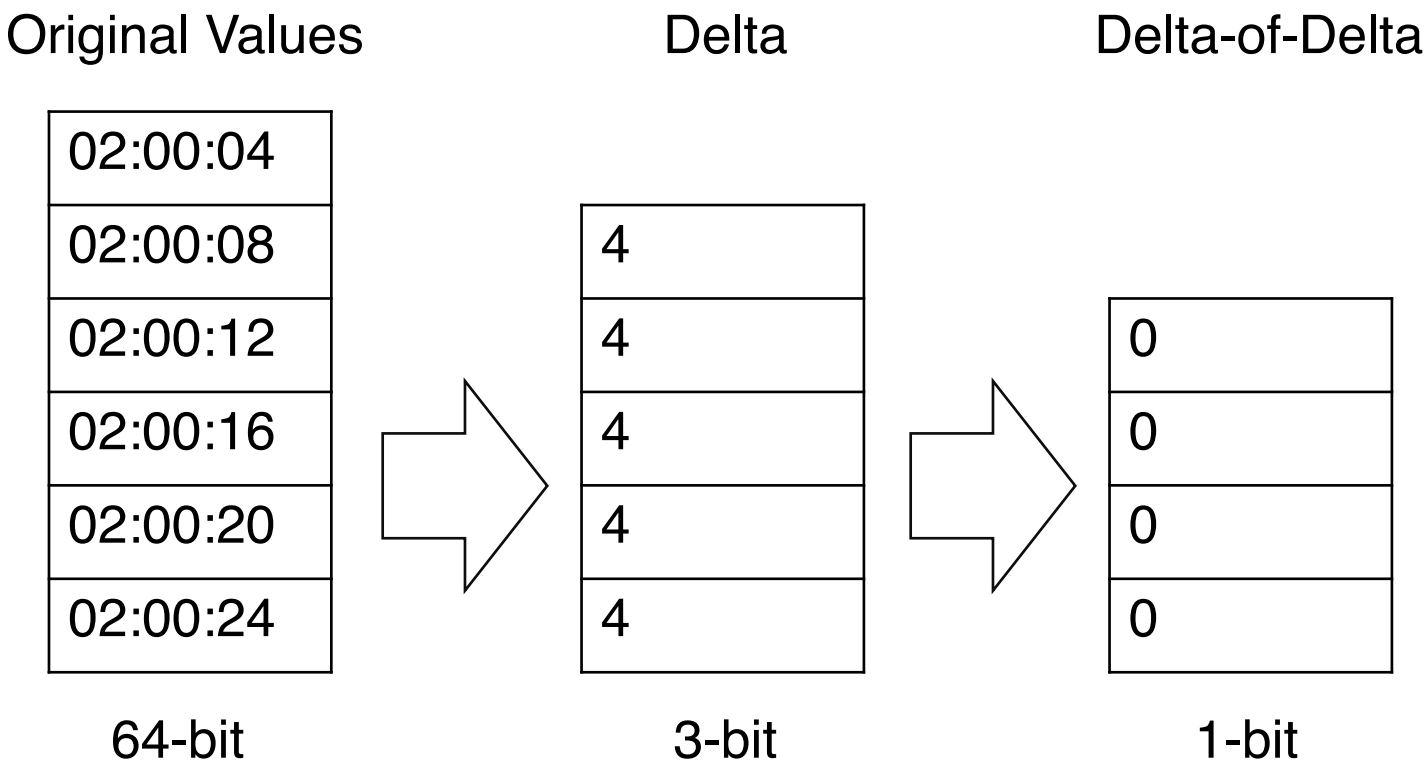
- XOR

Time Series Compression



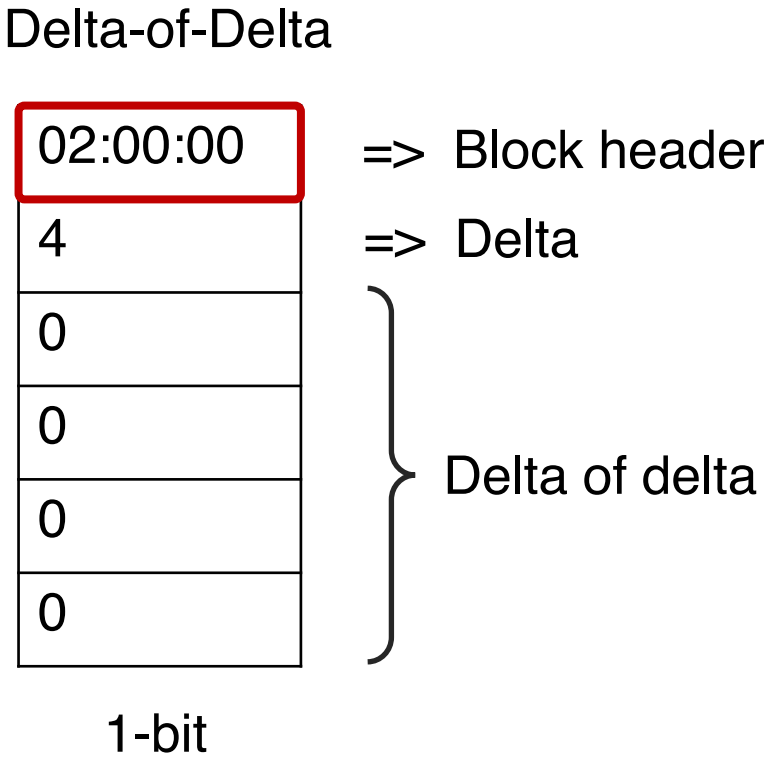
Timestamp Compression

Key observation: Vast majority of data points arrive at a fixed interval

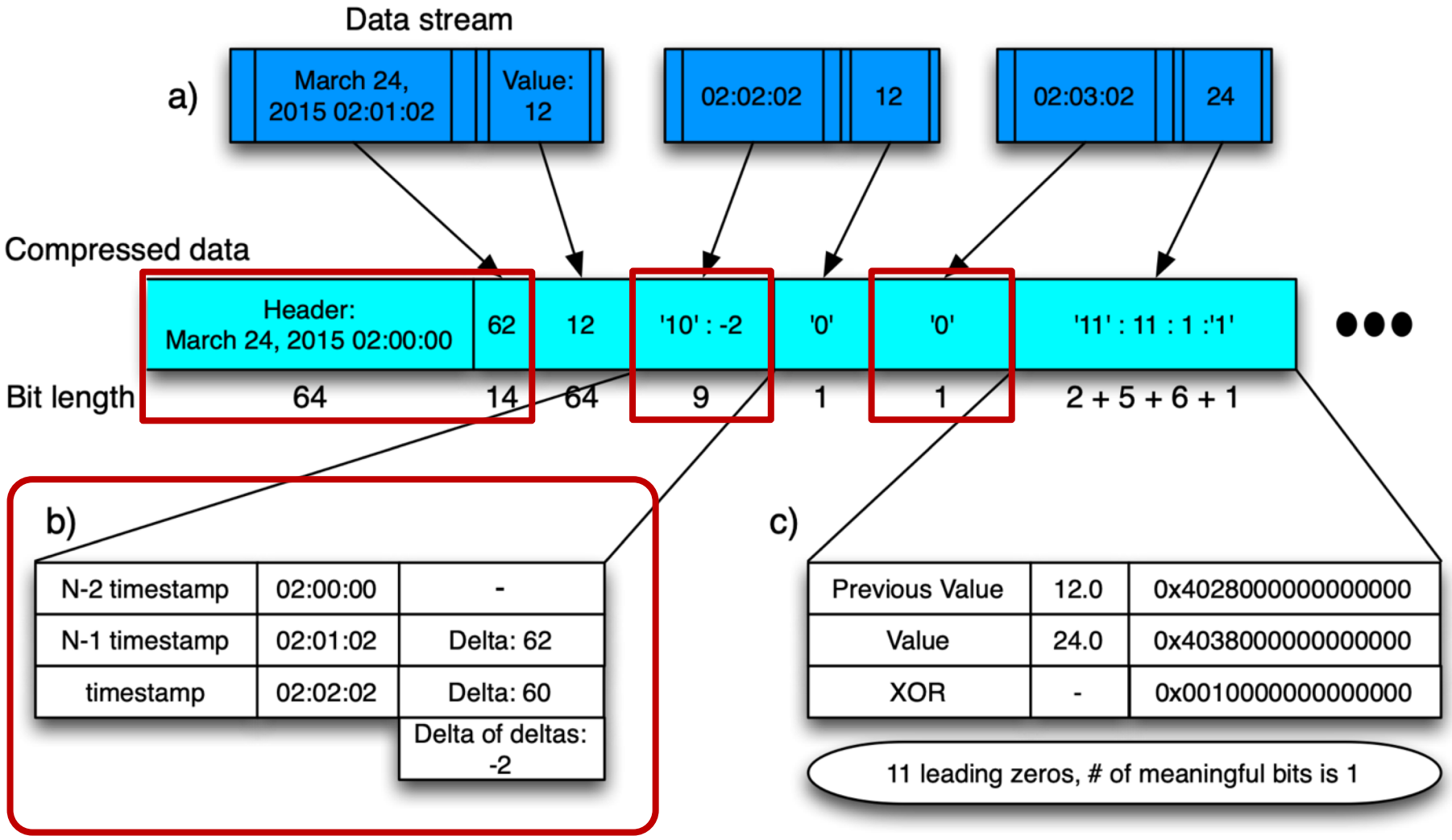


Timestamp Compression Algorithm

1. The block header stores the starting time stamp, t_{-1}, which is aligned to a two hour window; the first time stamp, t_0, in the block is stored as a delta from t_{-1} in 14 bits. [1]
2. For subsequent time stamps, t_n:
   (a) Calculate the delta of delta:
   $D = (t_n - t_{n-1}) - (t_{n-1} - t_{n-2})$
   (b) If D is zero, then store a single '0' bit
   (c) If D is between [-63, 64], store '10' followed by the value (7 bits)
   (d) If D is between [-255, 256], store '110' followed by the value (9 bits)
   (e) if D is between [-2047, 2048], store '1110' followed by the value (12 bits)
   (f) Otherwise store '1111' followed by D using 32 bits

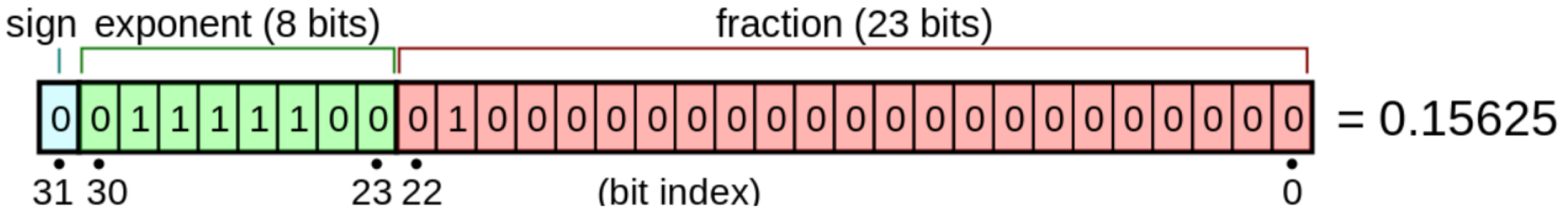


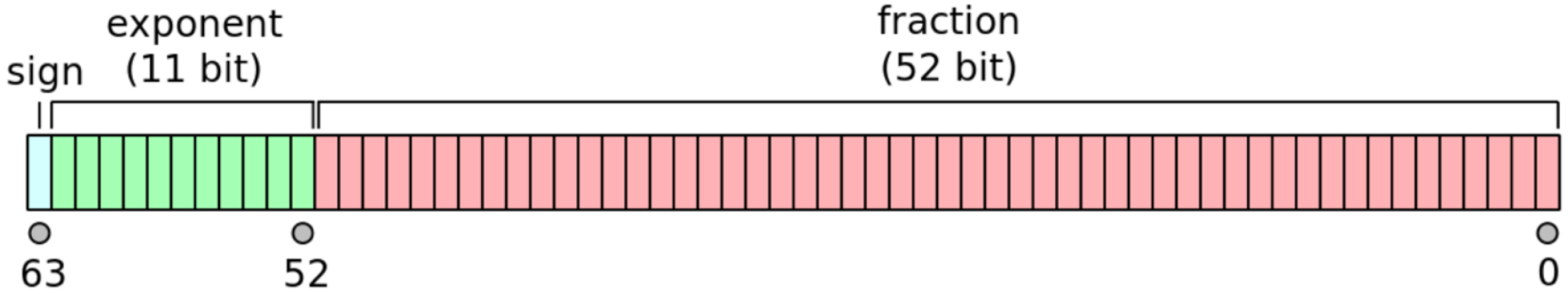
Timestamp Compression Example



Value Compression

Binary representation of floating point numbers





XOR truth table

Input		Output
A	B	
0	0	0
0	1	1
1	0	1
1	1	0

Value Compression – XOR

Decimal	Double Representation	XOR with previous
12	0x4028000000000000	
24	0x4038000000000000	0x0010000000000000
15	0x402e000000000000	0x0016000000000000
12	0x4028000000000000	0x0006000000000000
35	0x4041800000000000	0x0069800000000000

Decimal	Double Representation	XOR with previous
15.5	0x402f000000000000	
14.0625	0x402c200000000000	0x0003200000000000
3.25	0x400a000000000000	0x0026200000000000
8.625	0x4021400000000000	0x002b400000000000
13.1	0x402a333333333333	0x000b733333333333

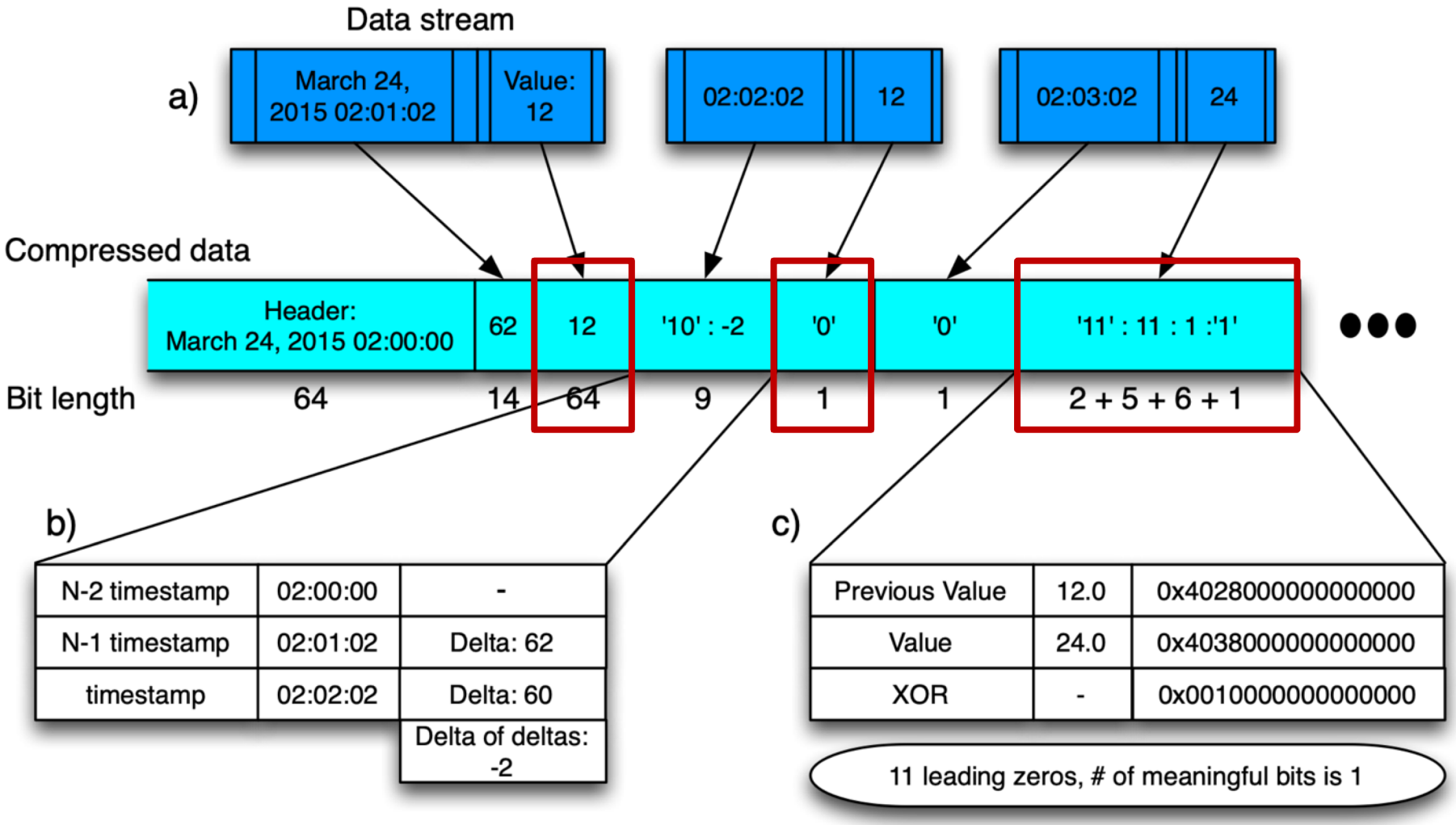
Value Compression – XOR

1. The first value is stored with no compression
2. If XOR with the previous is zero (same value), store single '0' bit
3. When XOR is non-zero, calculate the number of leading and trailing zeros in the XOR, store bit '1' followed by either a) or b):

   (a) (Control bit '0') If the block of meaningful bits falls within the block of previous meaningful bits, i.e., there are at least as many leading zeros and as many trailing zeros as with the previous value, use that information for the block position and just store the meaningful XORed value.

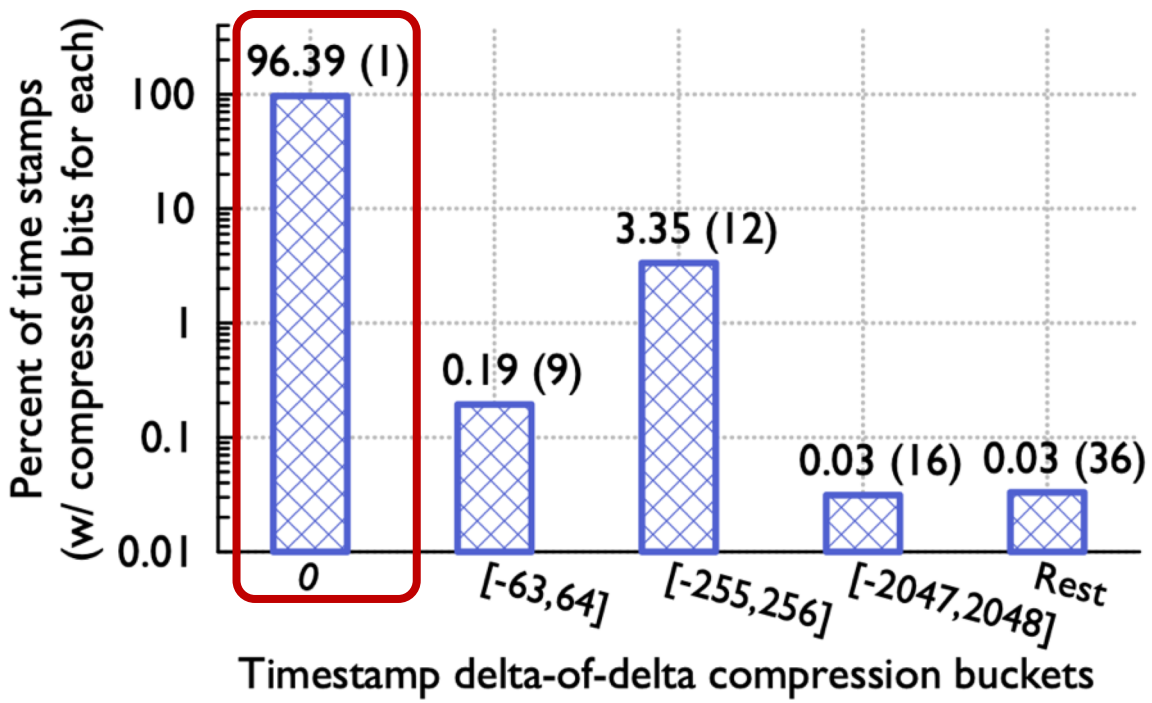
   (b) (Control bit '1') Store the length of the number of leading zeros in the next 5 bits, then store the length of the meaningful XORed value in the next 6 bits. Finally store the meaningful bits of the XORed value.

Value Compression Example

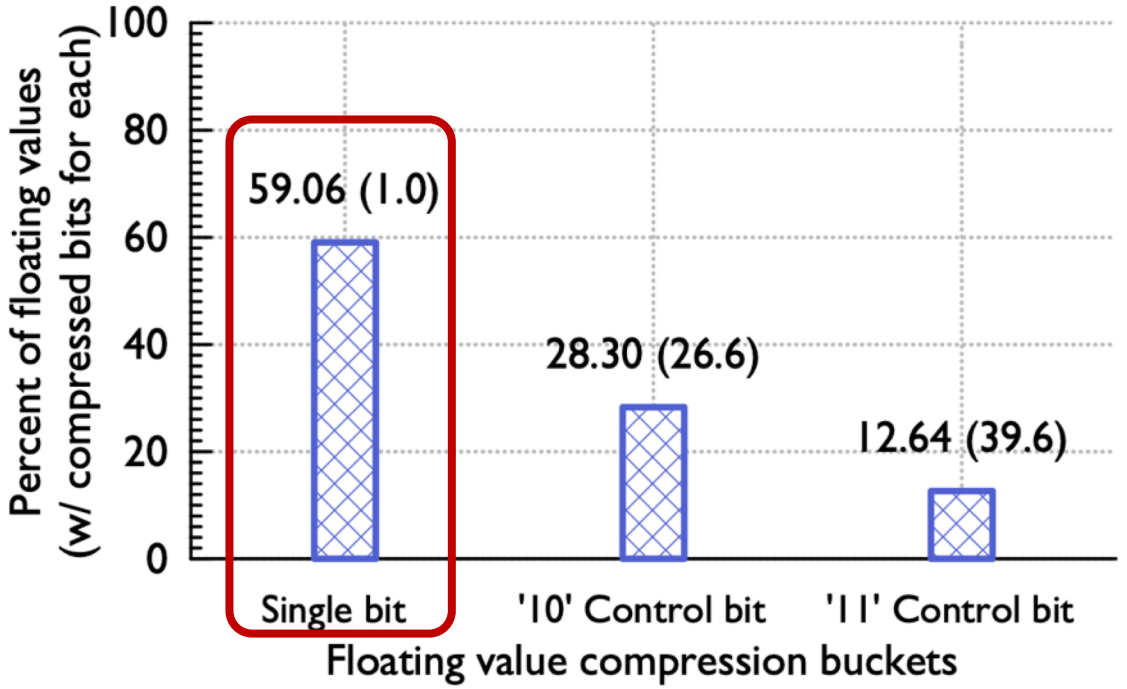


Compression Effectiveness

Timestamp Compression

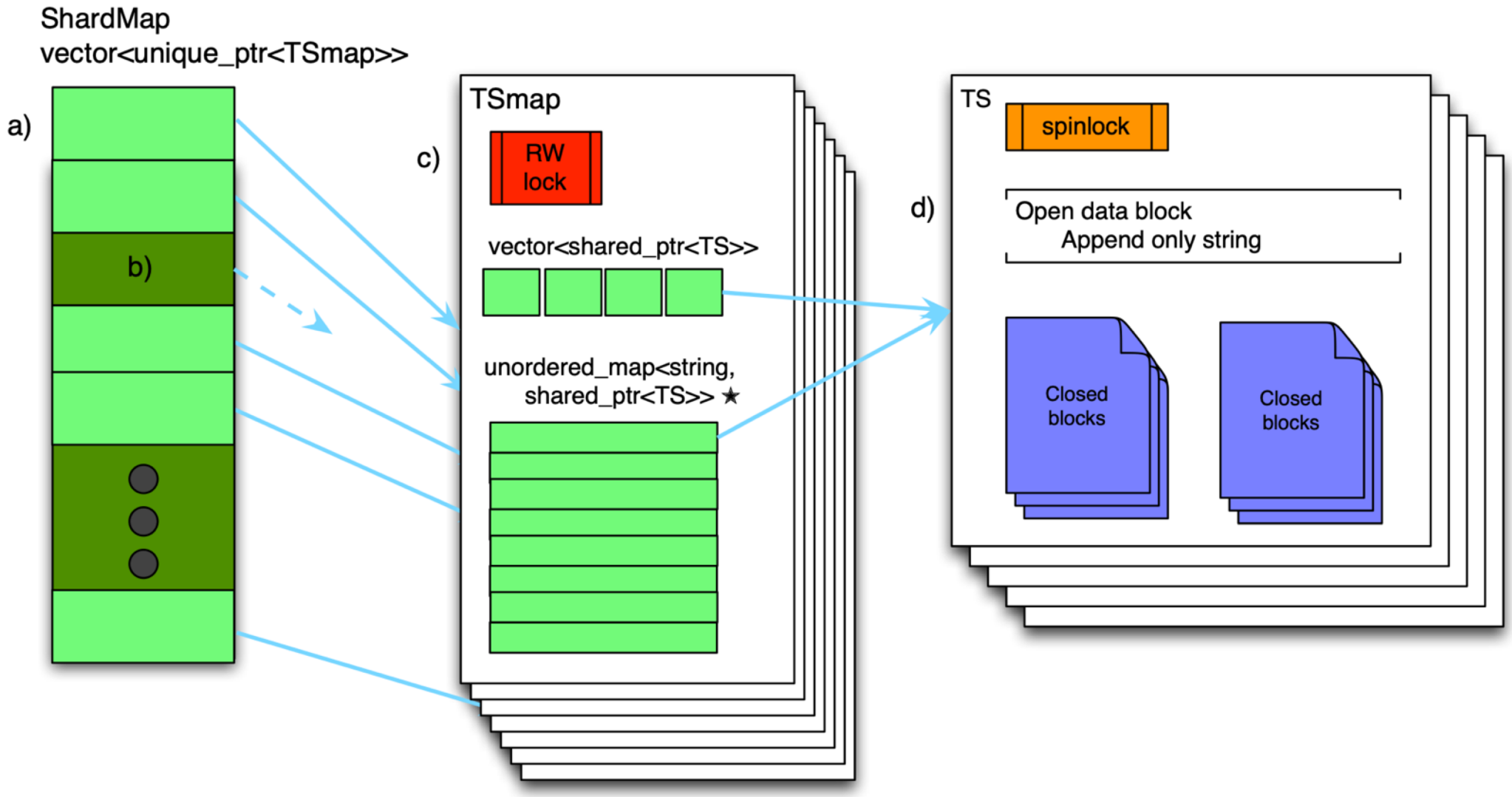


Value Compression



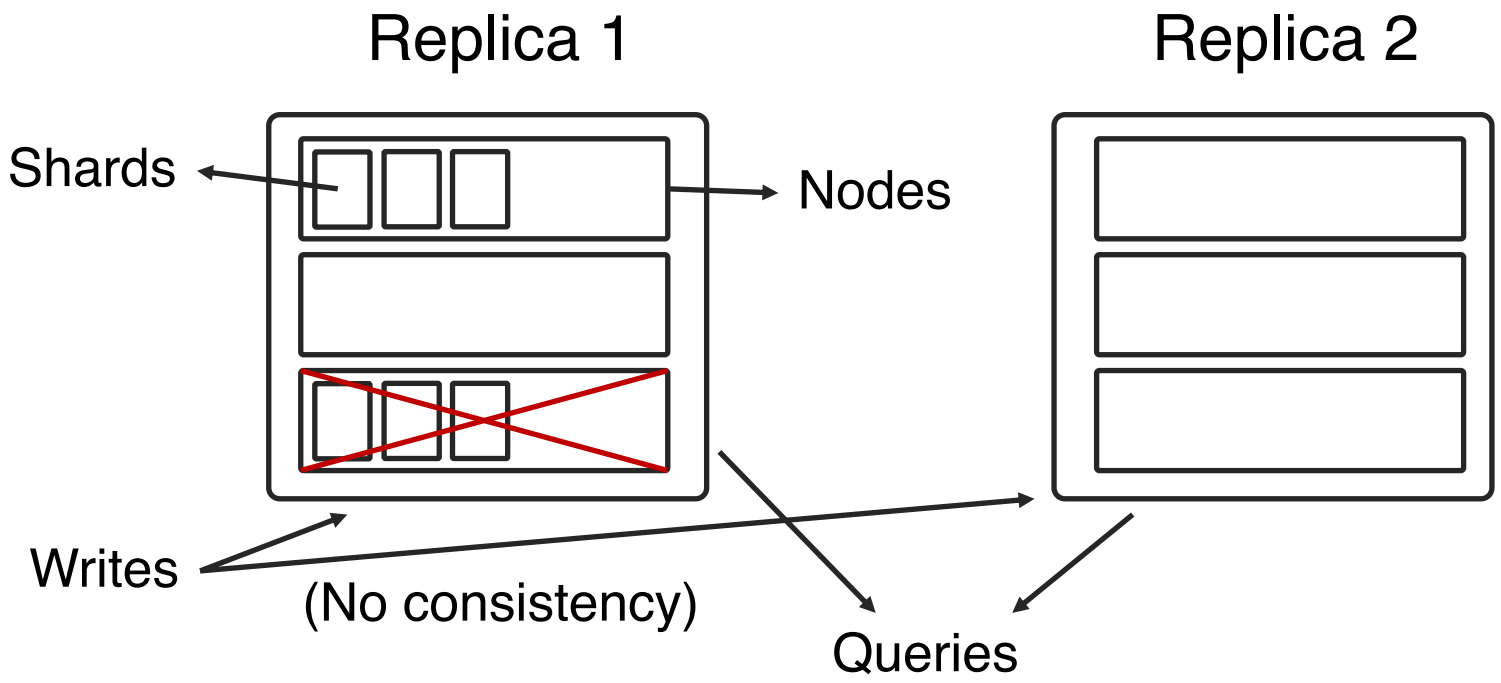
A two-hour block allows a compression ratio of **1.37 bytes per data point**

In-Memory Data Structure



★ TSmap uses a case-preserving, case-insensitive hash

Handling Failures

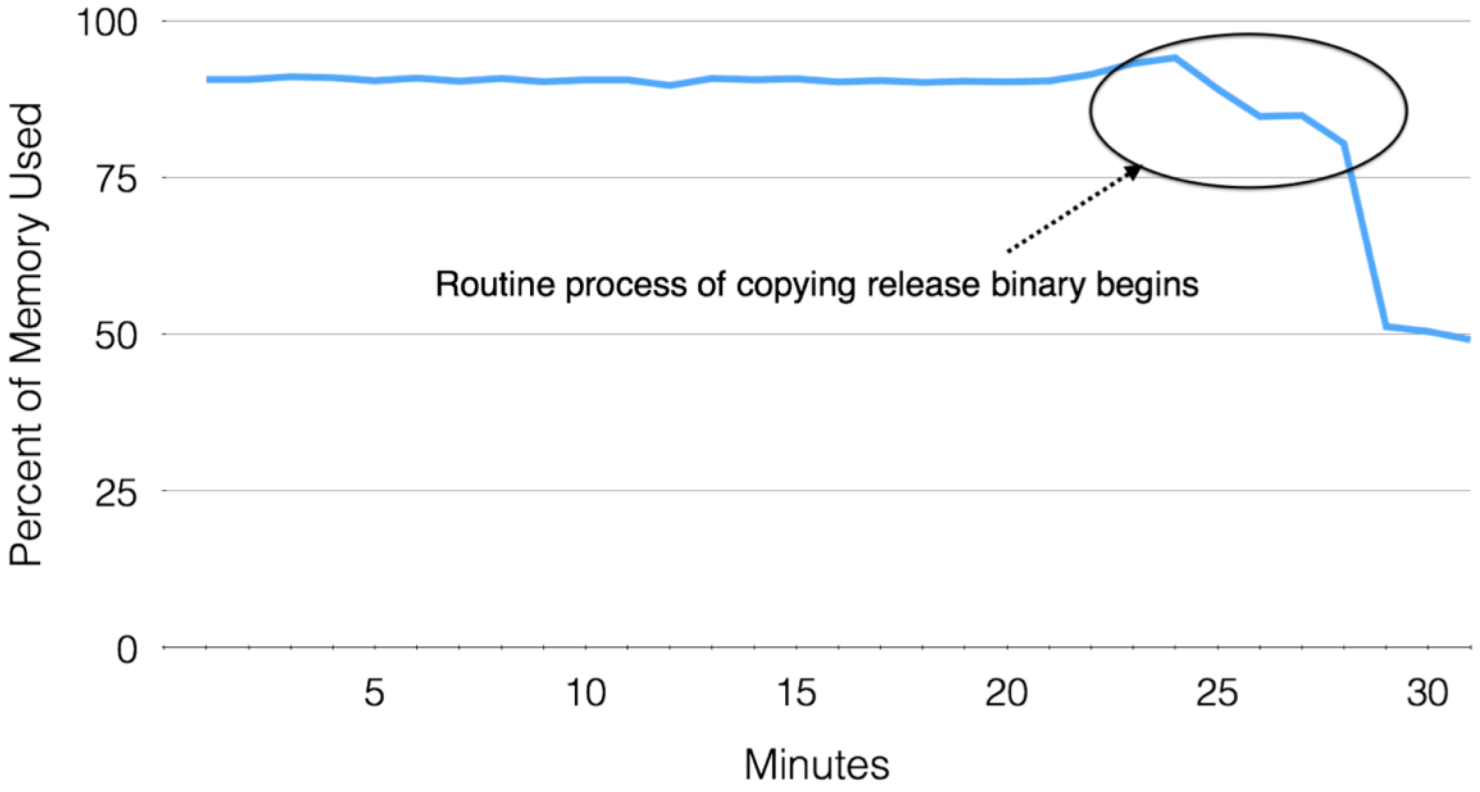


Node failure => data re-shard

- Data buffered in write client for 1 minute (prioritize recent data)

Query from a second replica if the first replica returns partial results

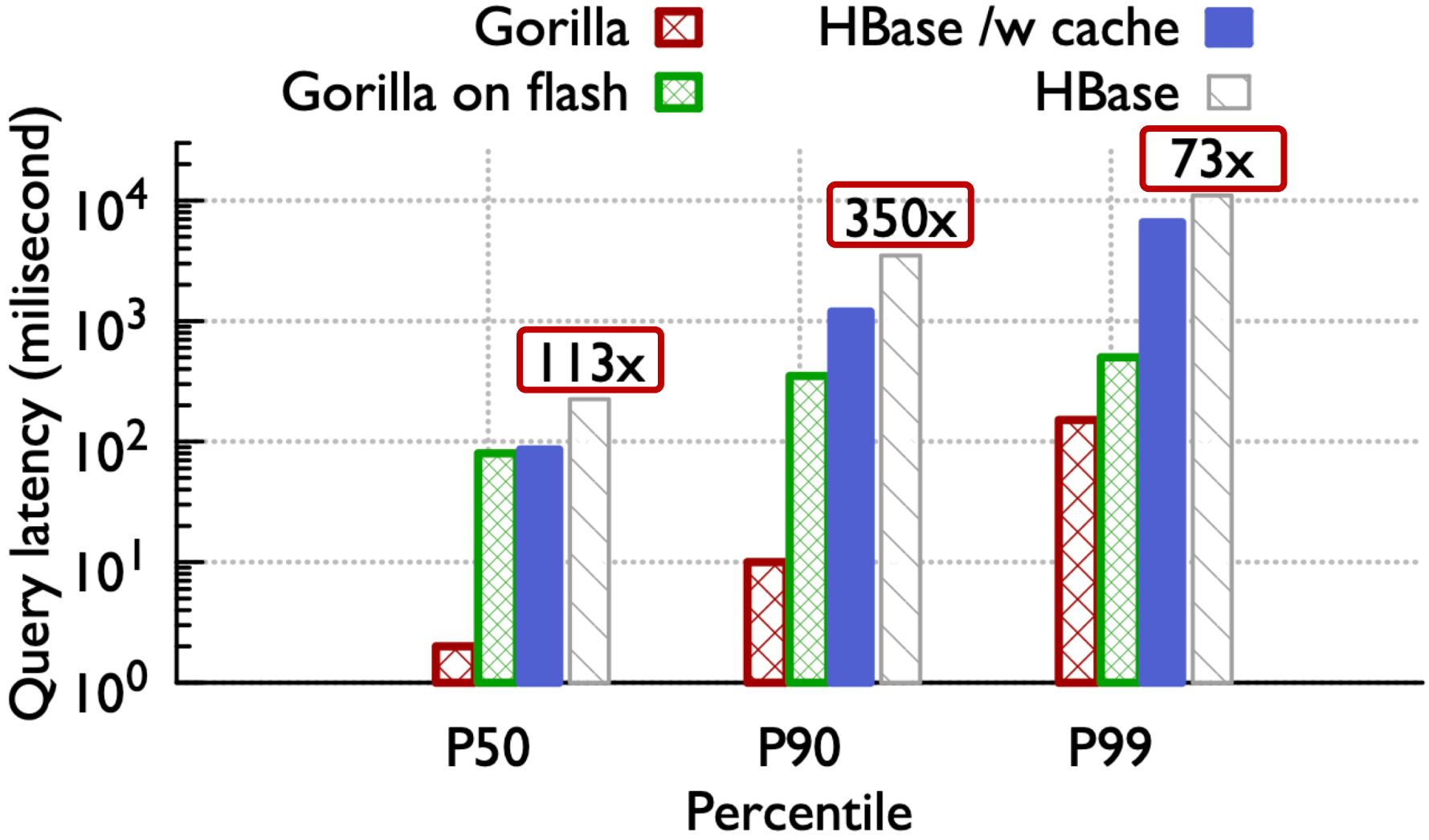
Example Use Case: Monitoring



Time series correlation search

- Routine process of copying the release binary to Facebook's web servers caused an anomalous drop in memory used across the site

Query Latency



Discussion

What's new in TSDB compared to Relational DB?

- Opportunities of data compression
- Different access pattern: append intensive
- Relaxed consistency model?
- Different query pattern

Time Series – Q/A

Conclusions in Gorilla generalized to other time series data?

Weak evaluation

Why store data up to 26 hours?

Gorilla works only with recent data

Other enterprise scale platforms implemented something like Gorilla?

Where else is TSDB useful?