Alibaba Cloud
Worldwide Cloud Services Partner

Two-Level Data Compression using Machine Learning in Time Series Database

Xinyang Yu, Yanqing Peng, Feifei Li,
Sheng Wang, Xiaowei Shen, Huijun Mai, Yue Xie

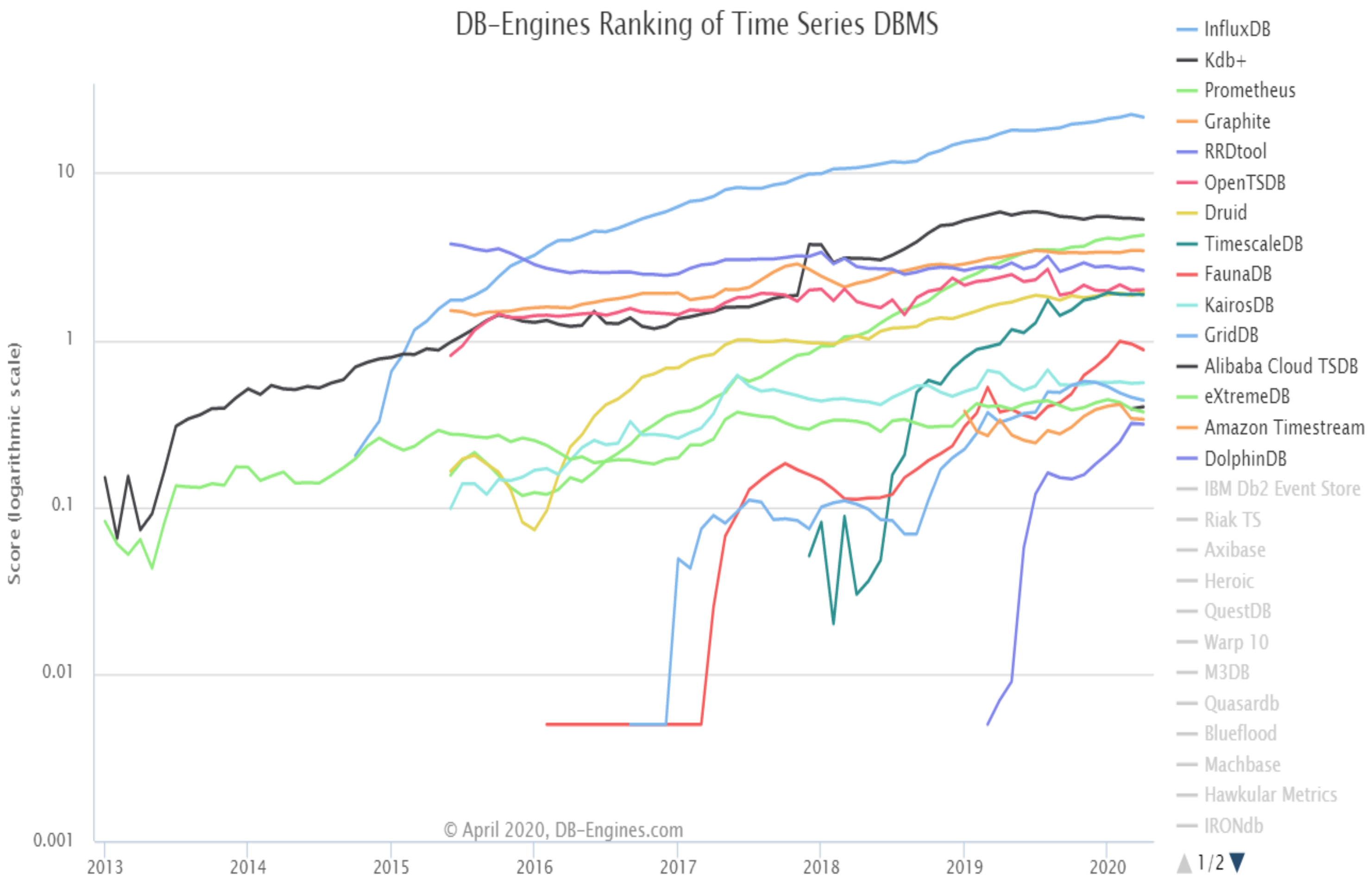
Agenda

- Background
- Two level compression framework
- Apply machine learning
- Results

Alibaba Cloud
Worldwide Cloud Services Partner

Background

Time series database status



*From **DB-Engines Ranking***

- Typical scenarios:
  - IoT and Sensor Monitoring
  - DevOps Monitoring
  - Real-Time Analytics

- Increasing significantly in the past few years

Major task

NASDAQ Composite Index

03:59	196.000	789	03:59	1207.140	1569	03:59	267.980	783
03:59	196.020	1048	03:59	1205.430	5604	03:59	267.930	9536
03:59	196.045	1366	03:59	1205.770	1573	03:59	267.910	4772
03:59	196.150	4212	03:59	1205.180	409	03:59	267.810	3917
03:59	196.082	4549	03:59	1204.995	1413	03:59	267.790	5635
03:59	196.120	360	03:59	1204.740	543	03:59	267.770	2112
03:59	196.130	2084	03:59	1204.740	106	03:59	267.735	3161
03:59	196.110	5195	03:59	1205.350	1571	03:59	267.920	2676
03:59	196.240	2338	03:59	1205.350	106	03:59	267.980	5020
03:59	196.100	1113	03:59	1205.350	7	03:59	267.940	3341
03:59	196.100	1557	03:59	1206.030	739	03:59	267.920	1497
03:59	196.220	861	03:59	1206.100	210	03:59	267.930	5588
03:59	196.220	402	03:59	1206.270	3177	03:59	267.980	1861
03:59	196.220	2110	03:59	1205.562	2997	03:59	267.980	180

- Top level user view
  - Query
  - Analyze
  - Predict
- Bottom level system view
  - Massive read, write
  - Compression, decompression
  - Downsample, Aggregation etc.
  - On <timestamp, value> 8B/8B

A key problem

Digital Economy

IoT era

5G network

…

Explosion of time series data

Apply Compression

Save footprint

Improve transfer latency

Improve process performance

How to compress efficiently?

Existing compression solutions

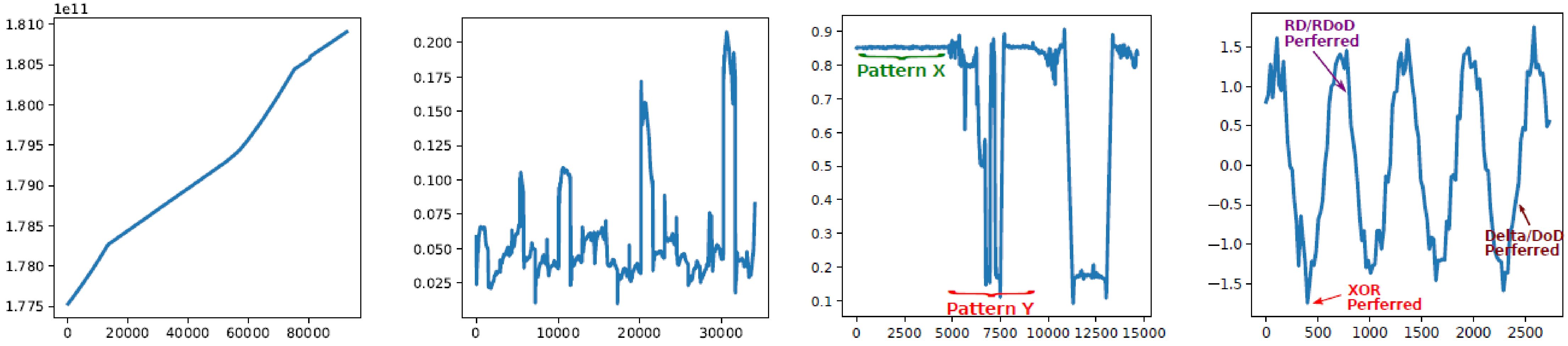
- Snappy: byte level prediction, RLE
- Gorilla: first apply delta-of-delta on timestamp and xor on value data
- MO: remove bit-level packing for parallel processing
- Sprintz: support predict, bit-packing, RLE and entropy coding
- …

- In general:
  - Support single mode, or a few static modes to compress the data
  - Most use bit-packing good for compression efficiency

Two level compression framework

Problem

Alibaba Cloud
Worldwide Cloud Services Partner



(a) Finance (b) Internet monitoring (c) Resource utilization (d) Sensing device

Fig. 1: Four use cases from real scenario; different use cases have different patterns. Fig 1c shows an example that different periods from data could have different patterns; Fig 1d illustrates the preferred compression scheme at different parts of data.

Three characteristics

Time Correlation

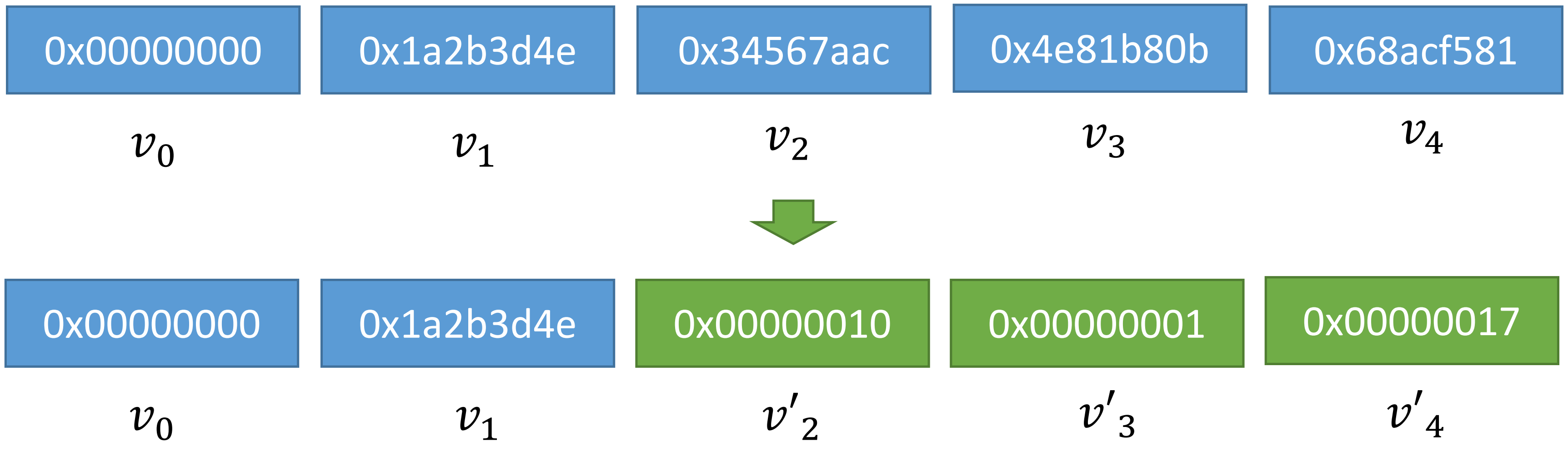
Pattern Diversity

Data Massiveness

Model Formalization: Transform

- Transform stage
  - Map raw data to transformed data that can be easily stored.
  - Capture the pattern of raw data.
  - 6 transform primitives; can be extended to fit other patterns.
  - Example: use delta-of-delta (DOD) on a near-linear time series data

Name	Desc.
Delta	$v_i - v_{i-1}$
RD	$v_{i-1} - v_i$
Xor	$v_i\ xor\ v_{i-1}$
DOD	$(v_i - v_{i-1})$-$(v_{i-1} - v_{i-2})$
RDOD	$(v_{i-1} - v_{i-2})$-$(v_i - v_{i-1})$
DX	$(v_i - v_{i-1})\text{xor}(v_{i-1} - v_{i-2})$



Model Formalization: Differential coding

- Differential coding:
  - Encode a value with less space by eliminating the zero bytes.
  - 3 coding primitives; can also be extended
  - Examples: Compress 5B data into 1B

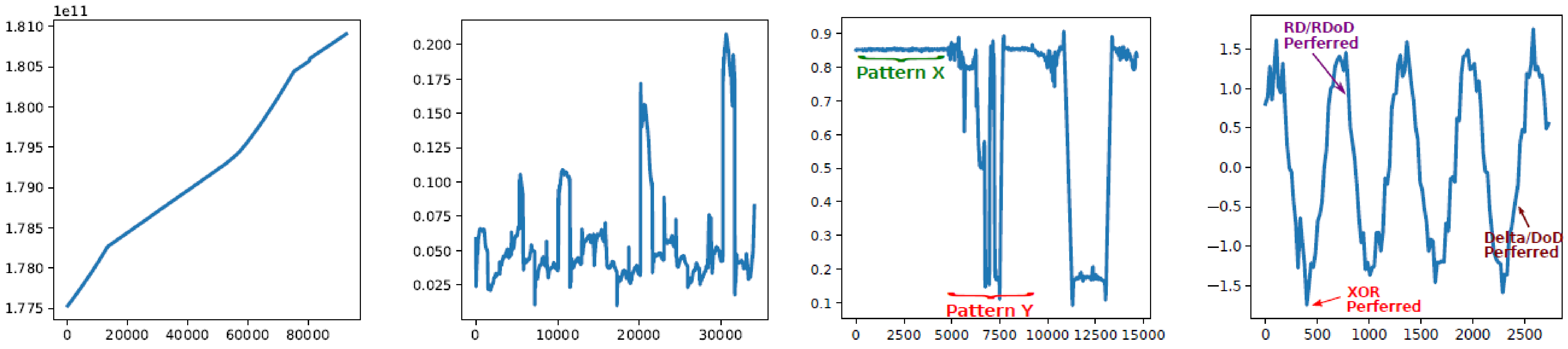
Primitive Name	Parameter	Format	Example		
			Input	Param	Output
`Offset`	`offByteShift`	1 byte: 1-bit control bits || 7-bit value 2 bytes: 2-bit control bits || 14-bit value 3 bytes: 3-bit control bits || 21-bit value	0x 00 17 00 00 00	3	control bits || 0b10111
`Bitmask`	`maskByteShift`	1 value: (up to 6-bit) bitmask || value 2 values: 6-bit bitmask || value1 || value2	0x 10 00 26 84 00	1	0b1011 || 0x 10 26 84
`Trailing-zero`	`N/A`	2-bit (or 3-bit) trailing-zero control bits || 3-bit non-zero control bits || value	0x 00 14 04 09 00	`N/A`	1 (0b01) || 3 (0b010) || 0x 14 04 09

Naïve adaptive compression

- Naïve solution: Try all possible combinations for each data point
- Problem: metadata explosion
  - We have to record the compression scheme for each data point
  - At least 2 Byte metadata (primitive choice + primitive parameter) for each data point
  - 25%+ overhead

Observation

- For most time series data,
  - The total number of different patterns is limited
  - Patterns can remain stable for a contiguous time range



(a) Finance (b) Internet monitoring (c) Resource utilization (d) Sensing device

Fig. 1: Four use cases from real scenario; different use cases have different patterns. Fig 1c shows an example that different periods from data could have different patterns; Fig 1d illustrates the preferred compression scheme at different parts of data.

Model Formalization: parameterized scheme space

TABLE IV: The control parameters in AMMMO which defines a compression scheme space in top level.

Parameter	Bits	Range	Description
majorMode	2	[0, 3]	Indicate the selected major mode
transType1	3	[0, 5]	Refer to a transform in Table III
transType2	3	[0, 5]	Similar to transType1
transType3	3	[0, 5]	Similar to transType1
offByteShift1	3	[0, 7]	Byte offset for 1-byte offset coding: offByteShift = offByteShift1
offByteShift2	1	[0, 1]	Byte offset for 2-byte offset coding: offByteShift = offByteShift1 − offByteShift2
offByteShift3	1	[0, 1]	Byte offset for 3-byte offset coding: offByteShift = offByteShift1 − offByteShift2 − offByteShift3
offUseSign	1	[0, 1]	Indicate if offset coding use sign
maskByteShift	3	[0, 5]	Byte offset for bitmask coding

All time series — Parameterized scheme space

Per timeline — Scheme space

$$S = \{s_1, s_2, \dots, s_n\}$$

Per point — Compression scheme | Compression scheme | $\cdots$

$$s = \langle a, b, \lambda_a, \lambda_b \rangle$$

$a \in P_{trans}$, $b \in P_{code}$, λ_a, λ_b *are parameters associated to a, b respectively*

- If supports 4 adaptive schemes, only 2 bits, 3.125% metadata overhead

Solution: 2 level compression framework

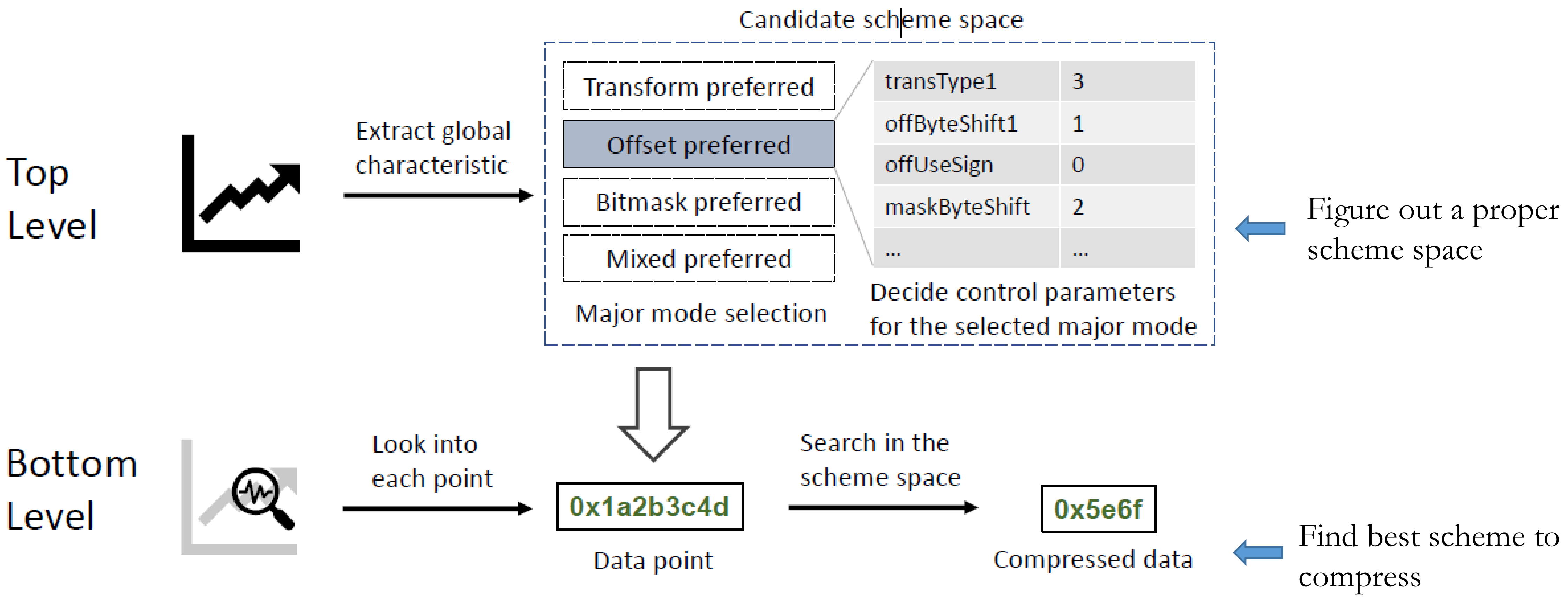


Fig. 2: The two-level compression framework AMMMO for compressing time-series data.

Rule-based scheme space selection

Algorithm 4: Rule-based Scheme Space Selection

Input: metric value sequence ms
Output: control parameter values $params$ in Table IV

```
/* PART I: calculate the benefit_score                          */
/* benefit_score: the total number of bytes can be
   saved against the worst case 9-byte original
   representation per point (i.e. 1 byte of control
   bits and 8 bytes of differential value) for
   different compress schemes in a timeline                     */
a 6 × 6 array with zero initializations: benefit_score;
for each point ms[i] where i ∈ [1, ms.length) do
    for each transform mode tm[j] in Table III do
        for each coding format cf[k] in Table II do
            benefit = calculateBenefit(ms[i], tm[j], cf[k]);
            benefit_score[j][k] += benefit;

/* PART II: calculate the params based on
   benefit_score                                                */
/* best_majorMode_score represents the best score
   among 4 major modes                                          */
best_majorMode_score = 0;
for each majorMode mm[i] where i ∈ [0, mm.length) do
    majorMode_score = 0;
    /* best_subMode_score represents the best score
       among 4 sub modes of the majorMode mm[i]                 */
    best_subMode_score = 0;
    for each subMode sm[j] where j ∈ [0, sm.length) do
        /* find the array indexes in benefit_score that
           match mm[i] and sm[j], say s and t                   */
        s, t = findIndex(mm[i], sm[j]);
        if benefit_score[s][t] > best_subMode_score then
            best_subMode_score = benefit_score[s][t];
    majorMode_score += best_submode_score;
    if majorMode_score > best_majorMode_score then
        params = findParams(mm[i]);

return params
```

- Use rules to select scheme space for the dataset.
- Problem:
  - Metric maybe not ideal
  - Human manually designed code
  - Not an automatic and adaptive method

Alibaba Cloud | Worldwide Cloud Services Partner

Apply machine learning

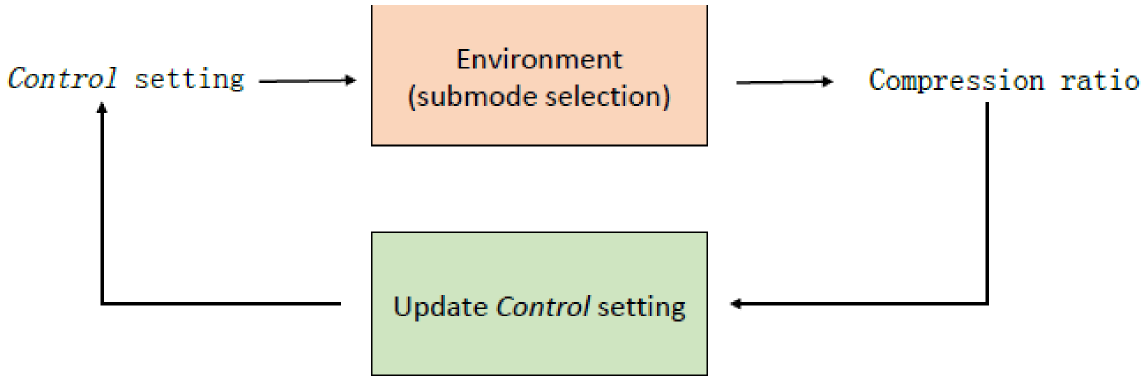
Use deep reinforcement learning

- Why machine learning?

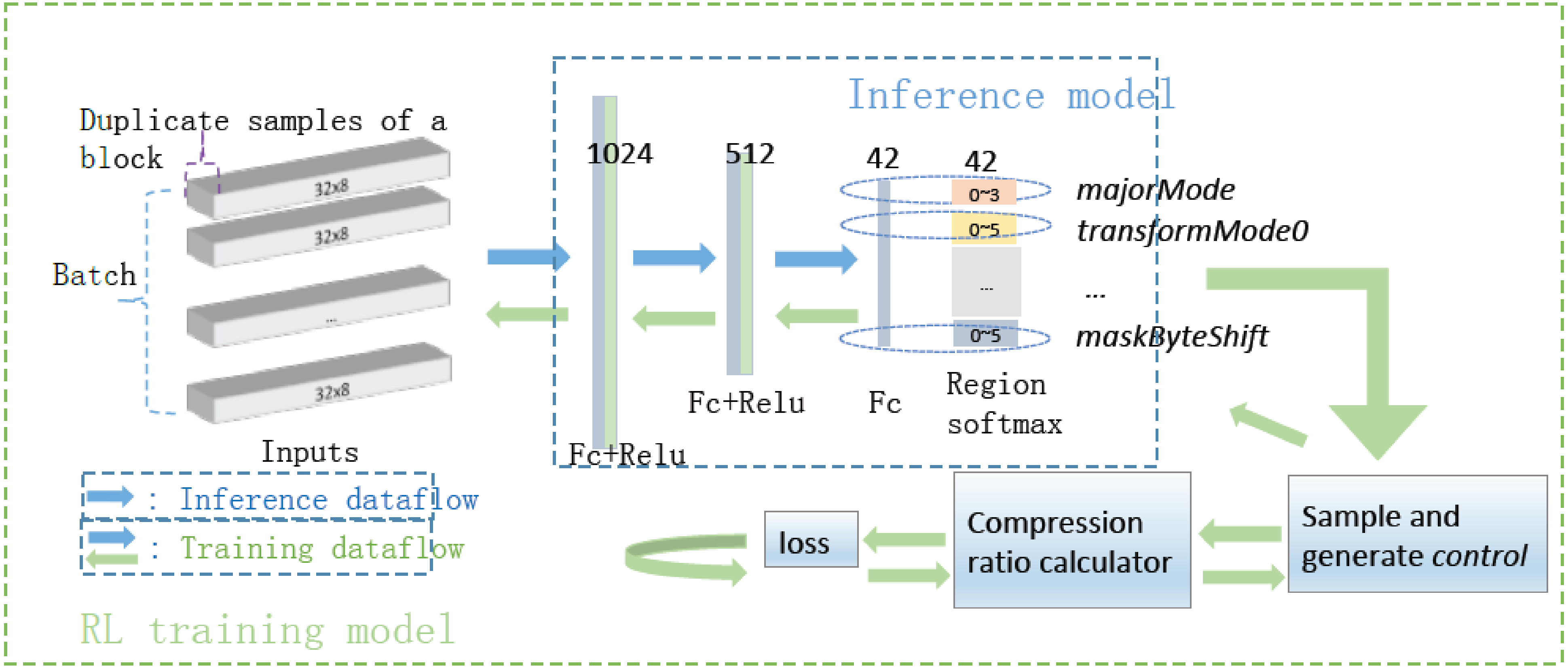
Parameter	Bits	Range
majorMode	2	[0, 3]
transType1	3	[0, 5]
transType2	3	[0, 5]
transType3	3	[0, 5]
offByteShift1	3	[0, 7]
offByteShift2	1	[0, 1]
offByteShift3	1	[0, 1]
offUseSign	1	[0, 1]
maskByteShift	3	[0, 5]

Multi-label classification problem

- Why reinforcement learning?
  - Not easy to create sample with label
    - 256^256 samples in theory, each need traverse 331,776 choices to figure out best
  - Not ideal one-class-label
  - Should be automatic



Neural network framework



- 32 points are taken as a basic block
- Duplicate and batch blocks to train, sample options, calculate loss and then do back propagation

$$\frac{1}{M*N}\sum_{i=1}^{M*N}(f_n(cop_i)*Hcs(cop_i)) - \lambda * H(cop) \qquad (6)$$

Results

Experiment setup

TABLE VII: Datasets with 28 selected time series.

Test Set *A*

Name	Points	Name	Points	Name	Points
IoT0	430,737	IoT6	430,413	Server35	147,395
IoT1	429,745	IoT7	313,539	Server41	136,594
IoT2	428,390	Server30	158,188	Server43	29,233
IoT3	344,581	Server31	147,385	Server46	154,585
IoT4	306,736	Server32	165,395	Server47	140,199
IoT5	372,868	Server34	140,194	Server48	157,051

Test Set *B*

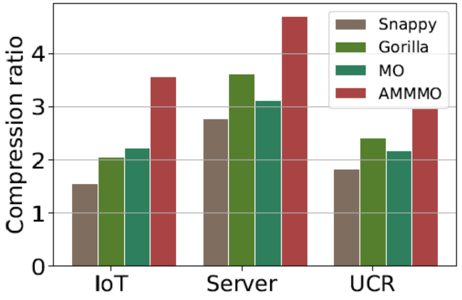
Name	Points	Name	Points
Server57	26,779	Server94	140,198
Server62	32,569	Server97	158,194
Server66	135,409	Server106	136,478
Server77	136,598	Server109	153,438
Server82	143,798	Server115	165,384

TABLE VIII: 8 longest time series datasets in UCR

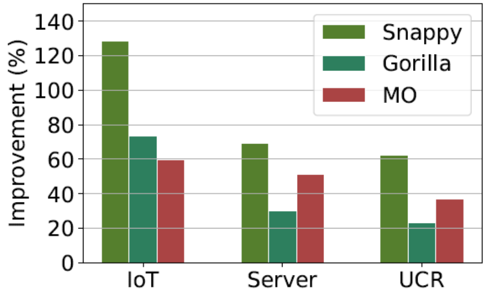
Test Set *A*		Test Set *B*	
Name	Points	Name	Points
HandOutlines	641,796	CinC_ECG_torso	8,190
Haptics	19,638	InlineSkate	16,929
StarLightCurves	155,496	MALLAT	6,138
UWaveGestureLibraryAll	115,168	Phoneme	4,092

- Gorilla: a state-of-the-art commercial bit-level compression algorithm applied in server side
- MO (Middle-Out): a public byte-level compression algorithm good for parallel processing
- Snappy: a general-purpose compression algorithm developed by Google
- AMMMO variants
  - AMMMO Lazy
  - AMMMO Rnd1000Avg
  - AMMMO Analyze
  - AMMMO ML
  - …

AMMMO performance comparison

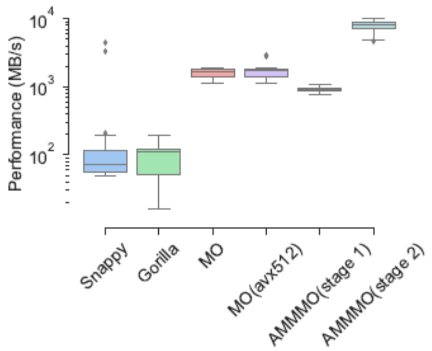


(a) Compression ratio

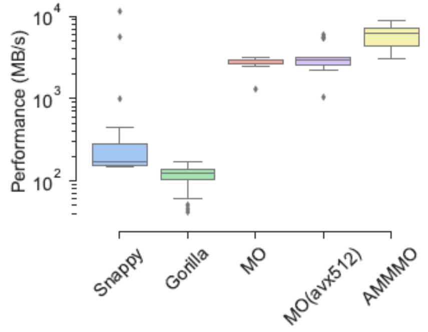


(b) Improvement of AMMMO over other schemes

Fig. 7: Comparison of compression ratios.



(a) Compression



(b) Decompression

Fig. 12: Compression and decompression performance comparison among different schemes.

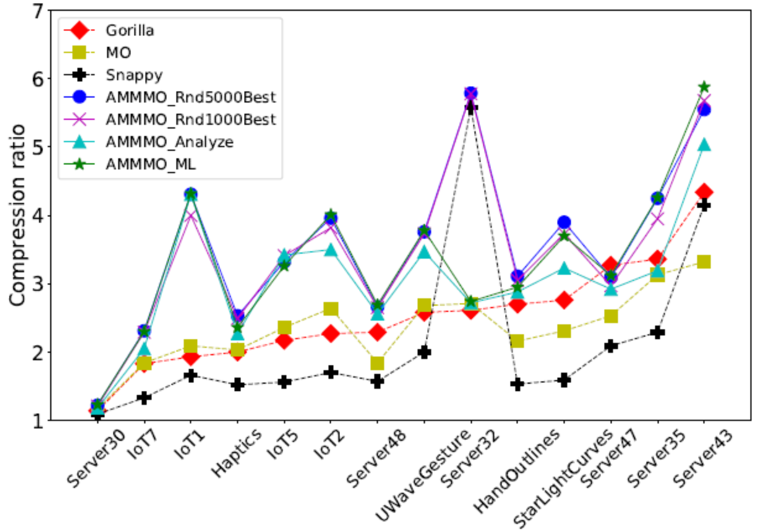


Fig. 8: Comparison of metric value compression ratios.

- Compression ratio:
  Snappy << Gorilla/MO << AMMMO
- AMMMO Compression efficiency:
  GB/s level in GPU platform

ML performance

Alibaba Cloud
Worldwide Cloud Services Partner

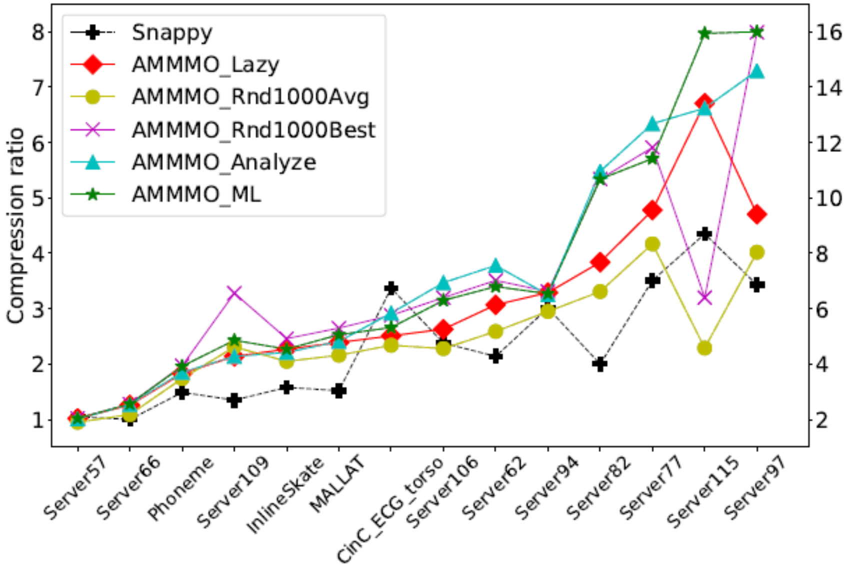


Fig. 10: Metric value compression ratios from different methods. Right vertical axis is for `Server97` and left axis for other datasets.

TABLE IX: Control settings selected by different AMMMO variants.

	Algorithm	major Mode	trans Type1	trans Type2	trans Type3	offByte Shift1	offByte Shift2	offByte Shift3	offUse Sign	maskByte Shift
IoT1	Analyze	3	0	5	0	3	0	0	0	1
	RandomBest	3	0	5	3	3	1	1	0	1
	ML	3	0	5	4	3	1	0	0	1
IoT2	Analyze	2	0	2	0	3	1	0	0	0
	RandomBest	2	5	0	3	3	1	0	0	0
	ML	2	0	0	5	3	1	0	0	0
Server35	Analyze	2	0	5	0	5	0	0	0	0
	RandomBest	3	4	2	4	5	0	0	1	5
	ML	3	3	2	5	5	0	0	0	5
Server48	Analyze	2	5	5	0	4	0	0	0	0
	RandomBest	3	0	5	4	6	1	1	0	4
	ML	3	4	5	0	4	0	0	0	4

- ML performs well in compression ratio view
- ML selects similar meaningful parameter value

Conclusion

- Proposed a two level framework for time-series data compression
  - In detail, we presents AMMMO definition, the result shows it achieves ~50% better compression efficiency, and fits parallel computing well

- Designed DRL logic to do scheme space selection (for final compression), which is an automatic, intelligent, and efficient way

Reference

References

[1] D. W. Blalock, S. Madden, and J. V. Guttag. Sprintz: Time series compression for the internet of things. IMWUT, 2(3):93:1 − 93:23, 2018.

[2] M. Burman. Time series compression library, based on the facebook’ s gorilla paper, 2016. https://github.com/burmanm/gorilla-tsc.

[3] H. Chen, J. Li, and P. Mohapatra. Race: time series compression with rate adaptivity and error bound for sensor networks. In 2004 IEEE International Conference on Mobile Ad-hoc and Sensor Systems, pages 124 − 133. IEEE, 2004.

[4] Y. Chen, E. Keogh, B. Hu, N. Begum, A. Bagnall, A. Mueen, and G. Batista. The ucr time series classification archive, 2015.

[5] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos. Fast subsequence matching in time-series databases. In Proceedings of the 1994 ACM SIGMOD international conference on Management of data, SIGMOD ’ 94, pages 419 − 429. ACM, 1994.

[6] S. Gunderson. Snappy: A fast compressor/decompressor, 2015.

[7] X. Guo, S. P. Singh, H. Lee, R. L. Lewis, and X. Wang. Deep learning for real-time atari game play using offline monte-carlo tree search planning. In NIPS, pages 3338 − 3346, 2014.

[8] B. Hawkins. Kairosdb: Fast time series database on cassandra.

[9] InuxDB. Open source time series, metrics, and analytics database, 2015.

[10] ITU-T and I. J. 1. Advanced video coding for generic audiovisual services, 2012. https://www.itu.int/rec/T-REC-H.264.

[11] S. K. Jensen, T. B. Pedersen, and C. Thomsen. Modelardb: Modular modelbased time series management with spark and cassandra. In Proceedings of the VLDB Endowment, volume 11. VLDB, 2018.

[12] S. D. T. Kelly, N. K. Suryadevara, and S. C. Mukhopadhyay. Towards the implementation of iot for environmental condition monitoring in homes. IEEE sensors journal, 13(10):3846 − 3853, 2013.

[13] E. J. Keogh, S. Lonardi, and C. A. Ratanamahatana. Towards parameterfree data mining. In KDD, pages 206 − 215. ACM, 2004.

[14] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In ICLR, 2015.

[15] I. Lazaridis and S. Mehrotra. Capturing sensor-generated time series with quality guarantees. In Proceedings of the 19th International Conference on Data Engineering, pages 429 − 440. IEEE, 2003.

[16] D. A. LELEWER and D. S. HIRSCHBERG. Data compression. In ACM Computing Surveys 19, volume 1 of ACM 1987, 1987.

References

[17] J. Lin, E. Keogh, S. Lonardi, and B. Chiu. A symbolic representation of time series, with implications for streaming algorithms. In Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery, pages 2 − 11. ACM, 2003.

[18] A. Mirhoseini, A. Goldie, H. Pham, B. Steiner, Q. V. Le, and J. Dean. A hierarchical model for device placement. In Proceedings of the International Conference on Learning Representations, ICLR ’ 18, 2018.

[19] A. Mirhoseini, H. Pham, Q. Le, M. Norouzi, S. Bengio, B. Steiner, Y. Zhou, N. Kumar, R. Larsen, and J. Dean. Device placement optimization with reinforcement learning. In International Conference on Machine Learning, ICML ’ 17, 2017.

[20] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In Proceedings of the 33rd International Conference on International Conference on Machine Learning, volume 48 of ICML’ 16, pages 1928 − 1937. ACM, 2016.

[21] OpenTSDB. A distributed, scalable monitoring system, 2013.

[22] D. Paul, Y. Peng, and F. Li. Bursty event detection throughout histories. In ICDE, pages 1370 − 1381. IEEE, 2019.

[23] T. Pelkonen, S. F. J. Teller, P. Cavallaro, Q. Huang, J. Meza, and K. Veeraraghavan. Gorilla: A fast, scalable, in-memory time series database. In Proceedings of the VLDB Endowment, volume 8 of VLDB’ 15, pages 1816 − 1827, Dec. 2015.

[24] P. Przymus and K. Kaczmarski. Dynamic compression strategy for time series database using gpu. In New Trends in Databases and Information Systems, pages 235 − 244. Springer, 2013.

[25] Schizofreny. Middle-out compression for time-series data, 2018.

[26] B. Schlegel, R. Gemulla, and W. Lehner. Fast integer compression using simd instructions. In Proceedings of the Sixth International Workshop on Data Management on New Hardware, volume 48 of DaMoN’ 10, pages 34 − 40. ACM, 2010.

[27] J. Shieh and E. Keogh. isax: disk-aware mining and indexing of massive time series datasets. Data Mining and Knowledge Discovery, 2009.

[28] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. G. Aja Huang, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis. Mastering the game of go without human knowledge. Nature, 550:354 − 359, Oct. 2017.

[29] E. Sitaridi, R. Mueller, T. Kaldewey, G. Lohman, and K. A. Ross. Massively-parallel lossless data decompression. In 2016 45th International Conference on Parallel Processing, ICPP ’ 16, pages 242 − 247. IEEE, 2016.

[30] G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand. Overview of the high efficiency video coding (hevc) standard. IEEE Trans. Circuits Syst. Video Technol., 22(12):1648 − 1667, Dec. 2012.

[31] F. D. E. Team. Rocksdb: A persistent key-value store for fast storage environments.

[32] R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. Machine Learning, 8(3):229 − 256, May 1992.

Thanks!
Q&A