Towards a Signature Based Compression Technique for Big Data Storage

Constantinos Costa
Rinnoco Ltd
Limassol, Cyprus
costa.c@rinnoco.com

Panos K. Chrysanthis
Rinnoco Ltd
Limassol, Cyprus
panos@rinnoco.com

Marios Costa
Rinnoco Ltd
Limassol, Cyprus
marios.c@rinnoco.com

Efstathios Stavrakis
Algolysis Ltd
Limassol, Cyprus
stathis@algolysis.com

Nicolas Nicolaou
Algolysis Ltd
Limassol, Cyprus
nicolas@algolysis.com

***Abstract*—Although the volume of stored data doubles every year, storage capacity costs decline only at a rate of less than 1/5 per year. At the same time, data is stored in multiple physical locations and remotely retrieved from multiple sites. Thus, minimizing data storage costs while maintaining data fidelity and efficient retrieval is still a key challenge in database systems. In addition to the raw big data, its associated metadata and indexes equally demand tremendous storage that impacts the I/O footprint of data centers. In this vision paper, we propose a new signature-based compression (SIBACO) technique that is able to: (i) incrementally store big data in an efficient way; and (ii) improve the retrieval time for data-intensive applications. SIBACO achieves higher compression ratios by combining and compressing columns differently based on the type and distribution of data and can be easily integrated with column and hybrid stores. We evaluate our proposed tool using real datasets showing that SIBACO outperforms "monolithic" compression schemes in terms of storage cost.**

***Index Terms*—signature based, compression, column stores, hybrid store**

I. Introduction

The ubiquitously available information sources and the advancements in data acquisition and processing techniques have led to an unprecedented increase in the data volumes. This huge amount of data enables smart data analysis, decision making and solutions, which transform the urban and work spaces and can be used to improve all aspects of life [1]–[3].

Effectively storing and processing big data workflows, such as spatiotemporal and telco big data (TBD), unlock a wide spectrum of smart applications, ranging from churn prediction of subscribers [4], city localization [5], 5G network optimization/user-experience assessment [1], [6], [7] and road traffic mapping [7] to name a few. If one considers that just one-hour Zoom group call requires between 360 MB and 1.2 GB of storage depending on the video quality [8], minimizing data storage costs and providing efficient retrieval is still a key challenge in database systems. In addition to the raw big data, its associated metadata and indexes equally demand tremendous storage that impacts the I/O footprint of data centers [9].

This work has been partially supported by Cyprus' Research Promotion Foundation RESTART programme, under project SIBACO INNOVOUCHERS/0722/0025. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of any of the sponsors.

Although the volume of electronically stored data doubles every year, storage capacity costs decline only at a rate of less than 1/5 per year [10]. However, the challenge of storing big data goes beyond the simplistic capacity cost calculated per GB. In order to have an impact or remain competitive, organizations and enterprises need to be able to: (i) store incremental big data in the most efficient manner; and (ii) improve the retrieval time for data-intensive analytics and exploration queries [11], [12]. These requirements are inherently conflicting, as it is well expressed in [13], where every data access system is described based on its Storage (read and write operations), Update tasks (i.e., storing data as they arrive incrementally), and Read tasks (i.e., data query and exploration).

To this end, optimized column stores were introduced to address the above requirements. Particularly, C-store was one of the first column-oriented database management systems, which uses aggressive compression techniques to reduce the storage cost [14]. It is very important to choose the correct encoding/compression scheme for a given data column to optimize the access to data stored on local or remote media [15]. Moreover, OLAP systems are using columnar data representations to exploit compression schemes based on dictionaries to minimize the storage cost [16]. These systems can yield high storage cost savings by using compression techniques that can adapt to the changes of data distribution alternating between local and global dictionaries [17].

In this vision paper, we present a novel signature-based compression technique, dubbed SIBACO, which enhances the current practice and can be easily integrated with column and hybrid stores to reduce further the storage cost. SIBACO's hypothesis is that multi-scheme data compression is more effective for complex big data by enabling incremental compression and partial decompression. Multi-scheme data compression uses different compression schemes that are more effective to be used for different columns based on their type and data characteristics. SIBACO breaks the data into groups of columns/rows and attempts to apply on individual or on a group of columns/rows the most suitable compression scheme. SIBACO selects a compression scheme for an individual or a group of columns by utilizing its knowledge base, which also maps a signature to a compression scheme, the database catalog and historical information.

We demonstrate the practicality of our SIBACO technique by experimenting with real datasets, namely Backblaze[1] and MovieLens[2], and analyzing them using data entropy. We show SIBACO's potential by comparing its efficiency with a sample of commonly used compression approaches from the *zipfile* library[3].

The rest of the paper is organized as follows: Section II presents the background and related work on the state-of-the-art compression techniques. Section III presents the overview of SIBACO and the details of each of its stage. Section IV presents our first experimental results and Section VI concludes our paper by discussing the next steps of our work.

II. Background and Related Work

As alluded above, SIBACO aims to support data-intensive applications, many of which need to be able to perform *exact* queries over stored data. Therefore we have first explored only *lossless* compression techniques.

We can divide the data lossless compression in the following four categories: (i) *Dictionary-based compression*, which uses a dictionary to keep track of the commonly occurring values in the data and replace them with a shorter code that is used as an index of the dictionary (e.g., LZ77, LZ78, LZW); (ii) *Statistical compression*, which uses statistical models to represent the probability of occurring values in the data (e.g., Huffman coding); (iii) *Transform-based compression* uses mathematical transformations to represent the data in a more compact form (e.g., Burrows-Wheeler Transform); and (iv) *Hybrid compression*, which uses a combination of methods from the previous three categories to achieve better compression ratios (e.g., DEFLATE, which combines LZ77 and Huffman coding, of the first two categories).

The lossless compression schemes used by large enterprises are being developed based on the requirements of their applications. For example, Google's Snappy[4], which uses byte-oriented operations and bit-stream encoding , and Facebook's Zstandard[5], which uses dictionary compression techniques, are suitable for real-time compression. Zstandard is configurable to allow the user to trade compression speed for higher compression ratios. Moreover, LZ4[6] is one of the fastest compression techniques that uses a byte-oriented LZ77 variant. In our experiments in this paper, we are using the publicly available compression algorithms provided by the *zipfile* library, which includes LZMA, BZIP2, and DEFLATE representing the compression categories ii, iii, and iv, respectively.

Compression in databases is being researched for more than three decades now. Due to OLAP over big data, column stores were introduced to address the huge volume and velocity of data requirements by integrating compression techniques to reduce the storage cost [14], [16], [18]. New compression-based optimizations emerged to minimize memory accesses that exploit CPU, RAM, and storage hardware advancements [19], [20]. Designing a compression-aware database management system can improve the compression ratio and speedup the queries further [21]. Moreover, novel solutions are leveraging machine learning techniques to efficiently transfer and store petabytes of data [11], [12], [22], [23].

In recent works related to SIBACO, data types and data organization have been exploited to speed up queries and save storage space using compression [15], [17], [24]. The use of partitioning data and separate applying a single compression technique can yield to higher compression ratios [25]. Opposite to our work, we advocate that use of different compression schemes can yield even better compression ratios. Even though our proposed solution falls more under the black-box category, it share similar ideas with white-box compression in the sense that it exposes to the applications the compression scheme via the database metadata [26].

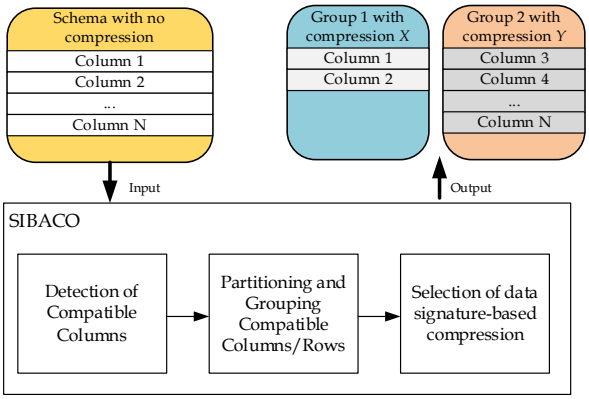

Fig. 1. SIBACO consists of three stages: (i) detection; (ii) partitioning and grouping; and (iii) selection. As an input, the current version of SIBACO accepts the uncompressed schema and outputs a set of partitioned column groups compressed with different compression schemes.

III. Overview

SIBACO achieves higher compression ratios by combining and compressing columns differently based on the type and distribution of the data in the different columns. Its compression process consists of three stages, as shown in Figure 1: (i) detection of compatible columns; (ii) partitioning and grouping of compatible columns; and (iii) then finally selection of data type-based compression. Subsequently, to retrieve the data SIBACO is using an ordered index that allows the retrieval of compressed files using the appropriate compression scheme.

The current version of SIBACO accepts as input a dataset and the compatible columns (first stage) and carries out the last two stages. Additionally, we use a naive partitioning/grouping and selection technique, i.e. entropy, based on empirical data.

[1]Backblaze: https://www.backblaze.com/b2/hard-drive-test-data.html
[2]MovieLens: https://grouplens.org/datasets/movielens/
[3]zipfile: https://docs.python.org/3/library/zipfile.html
[4]Snappy: https://google.github.io/snappy/
[5]Zstandard: https://facebook.github.io/zstd/
[6]LZ4: https://lz4.github.io/lz4/

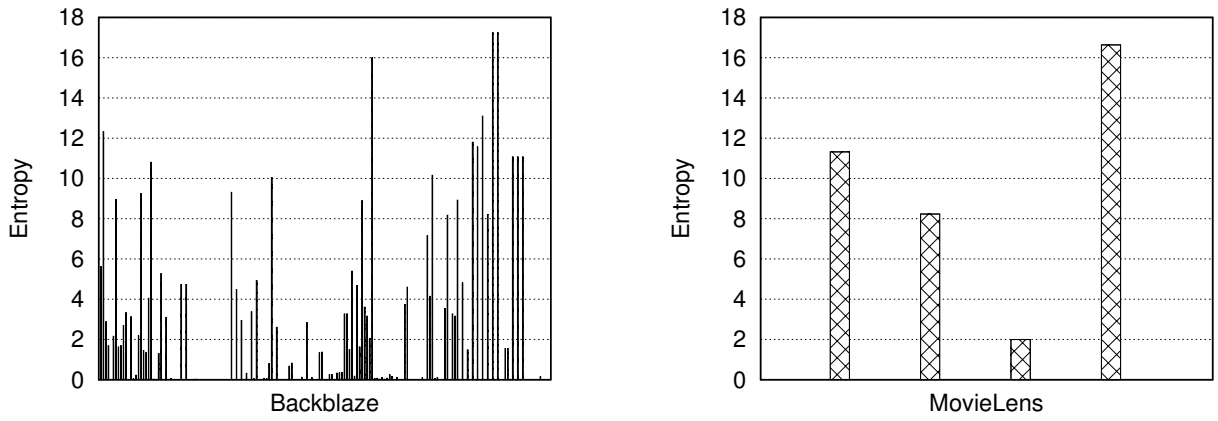

Fig. 2. The entropy of each attribute in the Backblaze dataset (left) and the MovieLens dataset (right).

TABLE I
THE TABLE WITH THE ENTROPY OF EACH ATTRIBUTE FOR THE MOVIELENS DATASET GENERATED BY THE FIRST STAGE

userId	movieId	rating	timestamp
11.31	8.22	1.99	16.63

A. Detection of compatible columns

In this first stage, the current version of SIBACO uses Shannon's source coding theorem [27] to identify if columns can be compressed yielding a significant reduction of storage. Shannon's entropy allows SIBACO to discover redundancy in the columns in a very efficient way, which enables the compression algorithms to encode the data using fewer bits. The output of this stage is a table containing the entropy of each column to be compression as shown in Table I for the MovieLens table.

B. Partitioning and grouping of compatible columns/rows.

The second stage partitions and groups the columns and rows that have similar signatures to achieve high compression ratios. The current version of this stage groups the columns with similar entropy from the table generated from the first stage. Particularly, SIBACO divides the columns into two groups using the *Group* function below. The first group consists of the columns $\{a_i, ..., a_n\}$ that are below the threshold $min(entropy) + \theta$, where θ is set based on empirical data (e.g., $\theta = 0.3$). The remaining columns with $entropy$ greater than $min(entropy) + \theta$ form the second group.

$$Group(a_i) = \begin{cases} a_i \in Set\ 1, \text{if } entropy(a_i) \leq min(entropy) + \theta \\ a_i \in Set\ 2, \text{otherwise} \end{cases}$$

C. Selection of data type-based compression

The final stage is to select the most appropriate compression algorithm based on the signature. In this stage, we are exploiting a knowledge base that was built based on a set different data characteristics (e.g., entropy, data distribution and types).

IV. EXPERIMENTAL EVALUATION

This section provides details regarding the algorithms, testbed, datasets, and metrics used for the preliminary evaluation of our SIBACO technique. In our experimentation, we chose the following algorithms because they are good representatives of the compression categories ii-iv and are readily available from the *zipfile* library.

Compression Algorithms:

- **DEFLATED**: uses a combination of LZ77 and Huffman coding.
- **BZIP2**: uses a combination of the Burrows-Wheeler transform and Huffman coding.
- **LZMA**: uses a combination of dictionary compression scheme, similar with LZ77, and arithmetic logic

Compared Techniques: Our aim in this experimental evaluation is to compare the following three techniques:

- **BASELINE**: This is the baseline ("monolithic") technique, which compresses the data being agnostic of the data characteristics. It compresses all the data with only a single compression scheme.
- **SIBACO-BASIC**: This is the basic approach of our proposed technique that uses a single compression scheme.
- **SIBACO**: This is our proposed technique that uses multiple compression schemes.

Datasets:

- **Backblaze**: This dataset is a subset of the publicly available hard drive metrics released by Backblaze. In our experiments we have only included data for the year 2022 (Q1-Q3). The data contains daily snapshots of more than 100.000 operational drives in a datacenter. The daily snapshot of each drive is represented by one row that captures basic drive metadata (i.e., serial number, device model, capacity) as well as SMART Attributes metrics. This dataset has 178 attributes and a total size of $\sim$20GB.

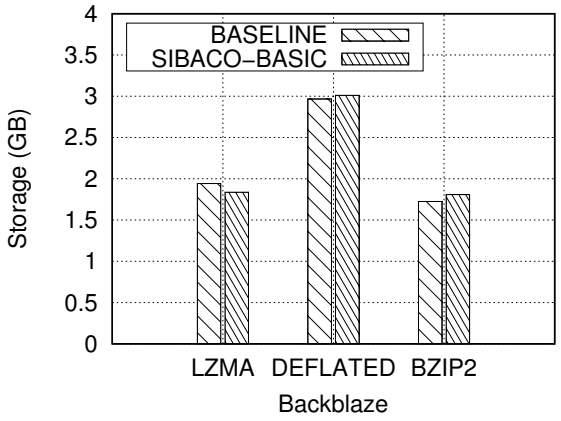

Fig. 3. We compare the total storage space of Baseline against SIBACO-BASIC using the Backblaze dataset and varying the compression algorithms.

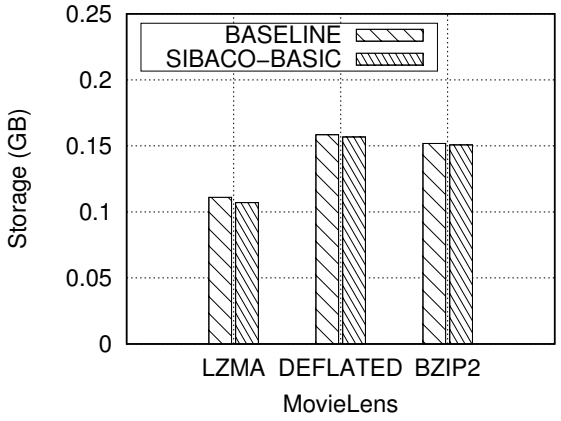

Fig. 4. We compare the total storage space of Baseline against SIBACO-BASIC using the MovieLens dataset and varying the compression algorithms.

- **MovieLens**: This dataset is a subset of the real-word MovieLens dataset collected by the GroupLens research laboratory. It contains 25 million ratings and one million tag applications applied to 62,000 movies by 162,000 users.This dataset has four attributes and and has a total size of ∼645MB.

Since we use entropy as part of SIBACO's compression signatures, we computed the entropy of the attributes included in the Backblaze and MovieLens datasets shown in Figure 2. Observing the Backblaze plot (Figure 2 (left)), it is clear that a lot of attributes have an entropy close to 0. This confirms that high compression ratios can be achieved. The 0 entropy attributes are usually optional attributes for future use, thus they are empty or not used, but still stored.

Testbed: Our evaluation is carried out on a low-end Linux server with 32 GB of DDR3 memory, a Intel Xeon E3-1226 v3 @ 3.30GHz processor with 4 cores and a 1TB SSD Drive.

V. Experimental Results

We conducted three experiments to assess the effectiveness of our SIBACO technique in reducing the storage. In this set of experiments we illustrate the potential of the first implementation of SIBACO by varying the compression schemes over two real datasets.

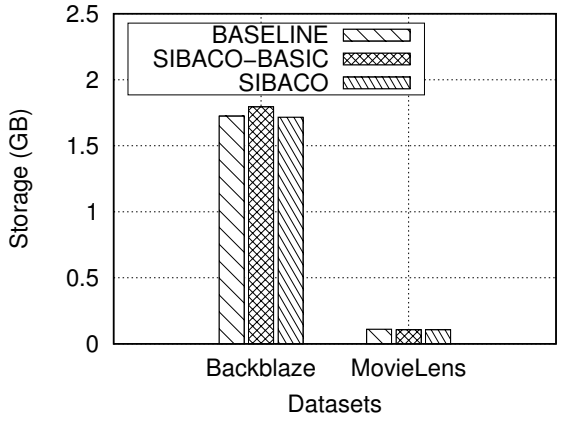

Fig. 5. We compare the baseline approach to SIBACO using the MovieLens dataset and varying the compression algorithms.

A. Experiment 1: SIBACO-BASIC vs Baseline using Backblaze

In this first experiment, SIBACO-BASIC splits 178 columns of the Backblaze dataset in two groups using $\theta = 0.3$. The first group consists of 73 columns with entropy close to 0, which was generated in the first stage of our technique. The second group consists of the remaining 105 columns.

SIBACO-BASIC outperforms the Baseline technique by up to ∼5% using LZMA for the Backblaze dataset as shown in Figure 3. LZMA and BZIP2 yield better compression than DEFLATED by up to ∼66% for both techniques.

This experiment indicates that the compression can affect how our proposed technique works by selecting the most appropriate compression scheme using the SIBACO knowledge base described in Section III.

B. Experiment 2: SIBACO-BASIC vs Baseline using MovieLens

In this second experiment, SIBACO-BASIC splits the four columns of the MovieLens dataset in two groups using $\theta = 0.3$. The first group consists of one column with the smallest entropy. The second group consists of the remaining three columns. Figure 4 shows that SIBACO-BASIC outperforms Baseline for all compression schemes by up to ∼4%.

C. Experiment 3: SIBACO vs Baseline using both datasets

In this third experiment, SIBACO (with multiple compression schemes) compares to the Baseline and SIBACO-BASIC approaches (that use a single compression scheme). SIBACO selects the best performing compression schemes for a group based on the previous experiments. As in the previous experiments, SIBACO splits the columns into two groups and applies the appropriate compression scheme for each group of the columns. Figure 5 shows that SIBACO achieves ∼4% and ∼0.5% better performance against BASELINE for MovieLens and Backblaze, respectively.

VI. Conclusion

In this vision paper, we pose that significant reduction in data storage can be achieved by using multi-scheme data compression and propose SIBACO that breaks the data into groups of columns and attempts to apply on individual groups the most suitable compression scheme. We describe the first SIBACO prototype that utilizes entropy to group columns and select the most appropriate compression scheme based on experimentally developed knowledge base.

In our preliminary experimental evaluation of SIBACO prototype using three compression algorithms and two real datasets, we observe reduction in storage space, up to $\sim$4% against the competitors. These first results are very encouraging since the current version of SIBACO does not utilize an extended knowledge base for compression signatures, and considers a limited number of compression scheme.

In our next steps, we aim to fully implement and refine all stages of our technique that can be easily plugged in storage systems, and develop a comprehensive knowledge base for compression signatures via analytical and experimental methods.

References

[1] C. Costa, G. Chatzimilioudis, D. Zeinalipour-Yazti, and M. F. Mokbel, "Efficient exploration of telco big data with compression and decaying," in *33rd IEEE International Conference on Data Engineering, ICDE*, 2017, pp. 1332–1343.

[2] ——, "SPATE: compacting and exploring telco big data," in *33rd IEEE International Conference on Data Engineering, ICDE*, 2017, pp. 1419–1420.

[3] D. Zeinalipour-Yazti and C. Claramunt, "COVID-19 mobile contact tracing apps (MCTA): A digital vaccine or a privacy demolition?" in *21st IEEE International Conference on Mobile Data Management, MDM*, 2020, pp. 1–4.

[4] Y. Huang, F. Zhu, M. Yuan, K. Deng, Y. Li, B. Ni, W. Dai, Q. Yang, and J. Zeng, "Telco churn prediction with big data," in *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '15, New York, NY, USA, 2015, p. 607–618.

[5] F. Zhu, C. Luo, M. Yuan, Y. Zhu, Z. Zhang, T. Gu, K. Deng, W. Rao, and J. Zeng, "City-scale localization with telco big data," in *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, ser. CIKM '16, New York, NY, USA, 2016, p. 439–448.

[6] A. P. Iyer, L. E. Li, and I. Stoica, "Celliq: Real-time cellular network analytics at scale," in *Proceedings of the 12th USENIX Conference on Networked Systems Design and Implementation*, ser. NSDI'15, USA, 2015, p. 309–322.

[7] C. Costa, G. Chatzimilioudis, D. Zeinalipour-Yazti, and M. F. Mokbel, "Towards real-time road traffic analytics using telco big data," in *Proceedings of the International Workshop on Real-Time Business Intelligence and Analytics, BIRTE*, 2017, pp. 5:1–5:5.

[8] OpenVault, "Pandemic impact on upstream broadband usage and network capacity," 2021.

[9] X. Ge and P. K. Chrysanthis, "On supporting scalable active learning-based interactive data exploration with uncertainty estimation index," in *Proceedings of the 24th International Conference on Extending Database Technology, EDBT*, 2021, pp. 421–426.

[10] D. C. Journal, "The cost of data storage and management: Where is it headed in 2016?" 2016.

[11] C. Costa, A. Konstantinidis, A. Charalampous, D. Zeinalipour-Yazti, and M. F. Mokbel, "Continuous decaying of telco big data with data postdiction," *GeoInformatica*, vol. 23, no. 4, pp. 533–557, 2019.

[12] C. Costa, A. Charalampous, A. Konstantinidis, D. Zeinalipour-Yazti, and M. F. Mokbel, "Decaying telco big data with data postdiction," in *19th IEEE International Conference on Mobile Data Management, MDM*, 2018, pp. 106–115.

[13] M. Athanassoulis, M. S. Kester, L. M. Maas, R. Stoica, S. Idreos, A. Ailamaki, and M. Callaghan, "Designing access methods: The RUM conjecture," in *Proceedings of the 19th International Conference on Extending Database Technology, EDBT*, 2016, pp. 461–466.

[14] M. Stonebraker, D. J. Abadi, A. Batkin, X. Chen, M. Cherniack, M. Ferreira, E. Lau, A. Lin, S. Madden, E. J. O'Neil, P. E. O'Neil, A. Rasin, N. Tran, and S. B. Zdonik, "C-store: A column-oriented DBMS," in *Proceedings of the 31st International Conference on Very Large Data Bases*, 2005, pp. 553–564.

[15] H. Jiang, C. Liu, J. Paparrizos, A. A. Chien, J. Ma, and A. J. Elmore, "Good to the last bit: Data-driven encoding with codecdb," in *SIGMOD '21: International Conference on Management of Data, Virtual Event*, 2021, pp. 843–856.

[16] D. J. Abadi, S. Madden, and M. Ferreira, "Integrating compression and execution in column-oriented database systems," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 2006, pp. 671–682.

[17] Y. Foufoulas, L. Sidirourgos, E. Stamatogiannakis, and Y. E. Ioannidis, "Adaptive compression for fast scans on string columns," in *SIGMOD '21: International Conference on Management of Data, Virtual Event*, 2021, pp. 554–562.

[18] S. Idreos, M. L. Kersten, and S. Manegold, "Self-organizing tuple reconstruction in column-stores," in *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD*, 2009, pp. 297–308.

[19] E. Choukse, M. Erez, and A. R. Alameldeen, "Compresso: Pragmatic main memory compression," in *51st Annual IEEE/ACM International Symposium on Microarchitecture, MICRO*, 2018, pp. 546–558.

[20] D. Habich, P. Damme, A. Ungethüm, J. Pietrzyk, A. Krause, J. Hildebrandt, and W. Lehner, "Morphstore - in-memory query processing based on morphing compressed intermediates LIVE," in *Proceedings of the 2019 International Conference on Management of Data, SIGMOD*, 2019, pp. 1917–1920.

[21] H. Kimura, V. R. Narasayya, and M. Syamala, "Compression aware physical database design," *Proc. VLDB Endow.*, vol. 4, no. 10, pp. 657–668, 2011.

[22] J. Zhang, G. Liu, D. Ding, and Z. Ma, "Transformer and upsampling-based point cloud compression," in *Proceedings of the 1st International Workshop on Advances in Point Cloud Compression, Processing and Analysis*, ser. APCCPA '22, New York, NY, USA, 2022, p. 33–39.

[23] A. Berezkin, A. Slepnev, R. Kirichek, D. Kukunin, and D. Matveev, "Data compression methods based on neural networks," in *ICFNDS 2021: The 5th International Conference on Future Networks & Distributed Systems*, 2021, pp. 511–515.

[24] M. Abebe, H. Lazu, and K. Daudjee, "Proteus: Autonomous adaptive storage for mixed workloads," in *SIGMOD '22: International Conference on Management of Data*, 2022, pp. 700–714.

[25] V. Raman and G. Swart, "How to wring a table dry: Entropy compression of relations and querying of compressed relations," in *Proceedings of the 32nd International Conference on Very Large Data Bases*, 2006, pp. 858–869.

[26] B. Ghita, D. G. Tomé, and P. A. Boncz, "White-box compression: Learning and exploiting compact table representations," in *10th Conference on Innovative Data Systems Research, CIDR*, 2020.

[27] C. E. Shannon, "A mathematical theory of communication," *Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, 1948.