Chimp: Efficient Lossless Compression of Floating Point Time Series Data

Panagiotis Liakos
panagiotisliakos@aueb.gr
Athens University of Economics and Business
Athens, Greece

Katia Papakonstantinopoulou
katia@aueb.gr
Athens University of Economics and Business
Athens, Greece

Yannis Kotidis
kotidis@aueb.gr
Athens University of Economics and Business
Athens, Greece

ABSTRACT

The increasing demands of diverse domains' applications generating and processing massive collections of *time series* data press the need for fast, streaming compression algorithms that effectively reduce the size of the data while also coping with the ingestion rate requirements of modern Time Series Management Systems. In this work we present Chimp, a novel streaming compression algorithm, suitable for floating point time series data. We empirically establish properties exhibited by a diverse set of time series and harness these features in our proposed encodings. Our experimental evaluation demonstrates that our approach *readily outperforms* competing techniques, attaining compression ratios that are on average around 50% of the space required by state-of-the-art streaming approaches. Moreover, our algorithm outperforms earlier techniques with regards to *(de)compression time*, offering a significantly *improved trade-off* between space and speed, ultimately providing significantly improved efficiency in storing and analyzing time series data.

1 INTRODUCTION

The size of high quality time series measurements produced at an unprecedented rate from applications found in multiple industries, raises significant challenges in terms of storing and harnessing this information to to deliver valuable and accurate insights. One way to cope with the increasing volume of data points is to compress them, to reduce the overall space requirements, while also providing data analysts with access to all historical data. Besides the obvious storage savings, the reduced requirements would also improve the performance of read queries, as fewer disk pages need to be read and more blocks can be cached in memory.

Indeed, Time Series Management Systems (TSMSs) [1] reduce the space requirements of storing measurements using *streaming* floating point compression approaches, such as Gorilla [4], an XOR-based floating point compression algorithm that is fast enough to handle the ingestion rate requirements of contemporary systems. However, despite the advantages and wide adoption of Gorilla, its respective space savings are very *modest* and, in any case, not on par with those of slower general purpose compression algorithms.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
GEC '23, June 26–27, 2023, Athens, GR
© 2023 Association for Computing Machinery.

In this paper we discuss Chimp, a novel lossless streaming compression algorithm that preserves the compression and decompression speed of earlier streaming approaches, while also providing significant space savings that are competitive with slower, yet extremely effective general purpose compression schemes. Chimp builds upon the distribution of the number of leading zeros to provide a very space efficient representation that is frequently reused by successive values. Moreover, we induce impressive compression rates by exploiting trailing zeros *only* when their number is large enough to provide savings. Finally, we exploit additional earlier values –besides the immediately previous one– to compare with the current value, to identify values that significantly boost our compression potential and help us greatly outperform the current state-of-the-art streaming approaches. Our solution offers space savings comparable to general-purpose compression algorithms that are up to 48x slower than Chimp. At the same time, Chimp is least as fast as the state-of-the-art streaming approaches. Our advancements in compression time, decompression time, and space requirements have substantial practical advantages for applications that work with time series data.

A full description of the Chimp algorithm along with extensive experiments over a wide range of large real-world datasets can be found in [3].

2 CHIMP ALGORITHM

An effective compression technique that is commonly used when dealing with floating point data is to perform a bitwise XOR operation between the current value and the previous value. The resulting set of bits is likely to contain a lot of leading zeros, as the sign and exponent are often identical for neighboring data points. XOR-based compression also attempts to exploit trailing zeros. However, long runs of trailing zeros are not very often. Thus, we can improve the space-efficiency of state-of-the-art approaches [4] by utilizing trailing zeros only when their number is large enough to provide savings. In particular, our proposed Chimp algorithm uses trailing zeros only when their number is larger than $log_2 64$. Moreover, we induce further savings by using the number of previous leading zeros only when it is the same with the current value, and by limiting the representation of the number of leading zeros to just 3 bits. The detailed operations of Chimp are portrayed with the diagram of Figure 1.

An important aspect that impacts the effectiveness of Chimp is the similarity of consecutive values. In the case of identical values, Chimp spends only two flag bits to represent the second instance. Equivalently, when the similarity between the values causes a XORed result with many leading and trailing zeros, Chimp is able to reduce

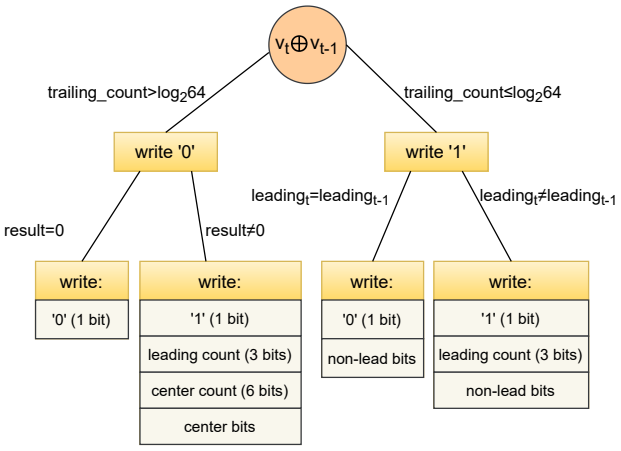


Figure 1: Operations of the Chimp algorithm.

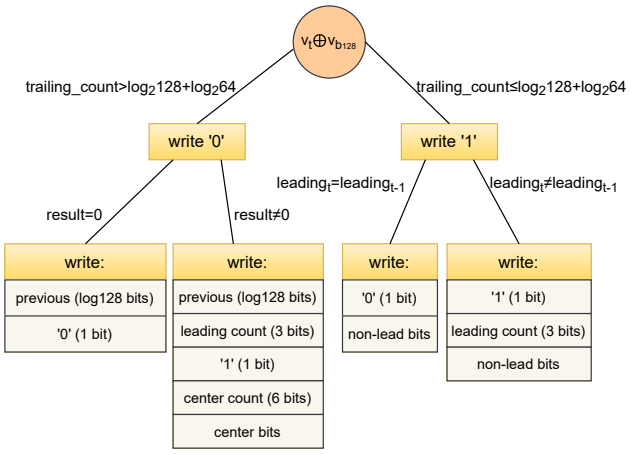


Figure 2: Operations of the Chimp$_{128}$ variation.

the overall space requirements significantly. However, our investigation indicates that such cases are not very common, especially when working with measurements of high precision. However, we find that there is great potential with regard to finding similar measurements when exploiting more than one previous values. The improvement is evident in most time series when using just 16 previous values. Setting this number to 128, results in very few cases of XORed values with less than 6 trailing zeros for all the time series of our dataset. On the contrary, we very often find a large number of trailing zeros, if not an identical value.

The diagram of Figure 2 illustrates the encoding scheme of Chimp$_{128}$, that makes use of 128 previous values, instead of a single one. The bitwise XOR operation in Chimp$_{128}$ is performed between the current value v_t and the best of 128 previous values in terms of most trailing zeros, $v_{b_{128}}$. If the resulting number of trailing zeros *surpasses* the number of bits needed to denote the previous value used ($log_2 128$ bits) plus the number of bits required to specify the number of meaningful bits ($log_2 64$ bits), then we make use of and actually store the previous value used (two bottom-left cases of Figure 2). Otherwise, the use of $v_{b_{128}}$ is not particularly useful and, due to the flexibility of Chimp, we can use the immediately previous value v_{t-1} instead, and avoid wasting additional bits to denote the previous value used (two bottom-right cases of Figure 2).

3 EXPERIMENTAL EVALUATION

We compare our Chimp algorithm and its Chimp$_{128}$ variation against 5 general purpose and 2 streaming floating point compression algorithms. Figure 3 illustrates the average trade-off between compression time and ratio achieved for the 14 time series of our dataset. We see that Chimp surpasses the speed of streaming approaches while also providing reduced space requirements. Chimp$_{128}$ is equivalently fast and offers space savings that are competitive with significantly slower general purpose compression algorithms. The evident superiority of Chimp$_{128}$ as demonstrated in Figure 3 establishes our approach as the preferred option for compressing floating-point values in the domain of time series data.

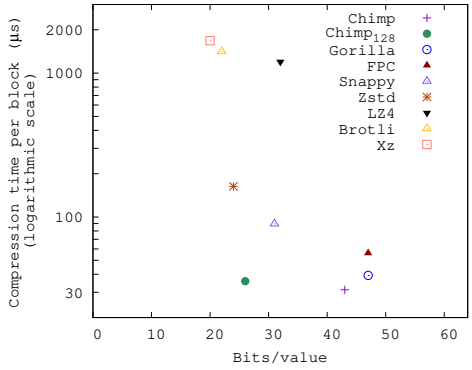


Figure 3: Trade-off between compression time (μs in log scale) and space (bits/value) for various algorithms.

4 OPEN DIRECTIONS

We plan to integrate Chimp with lossy streaming compression techniques [2] to enhance their compression by efficiently encoding floating-point values.

ACKNOWLEDGMENTS

This work has received funding from the Hellenic Foundation for Research and Innovation (HFRI) and the General Secretariat for Research and Technology (GSRT), under grant agreement No 779.

REFERENCES

[1] Søren Kejser Jensen, Torben Bach Pedersen, and Christian Thomsen. 2017. Time series management systems: A survey. *IEEE Trans. Knowl. Data Eng.*, 29, 11, 2581–2600. DOI: 10.1109/TKDE.2017.2740932.

[2] Xenophon Kitsios, Panagiotis Liakos, Katia Papakonstantinopoulou, and Yannis Kotidis. 2023. Sim-piece: highly accurate piecewise linear approximation through similar segment merging. *Proc. VLDB Endow.*, 16, 8, 1910–1922.

[3] Panagiotis Liakos, Katia Papakonstantinopoulou, and Yannis Kotidis. 2022. Chimp: efficient lossless floating point compression for time series databases. *Proc. VLDB Endow.*, 15, 11, 3058–3070.

[4] Tuomas Pelkonen, Scott Franklin, Paul Cavallaro, Qi Huang, Justin Meza, Justin Teller, and Kaushik Veeraraghavan. 2015. Gorilla: A fast, scalable, in-memory time series database. *Proc. VLDB Endow.*, 8, 12, 1816–1827. DOI: 10.14778/2824032.2824078.