Indian Journal of Information Sources and Services
ISSN: 2231-6094 (P) Vol.15, No.4, 2025, pp.169-176

DOI: https://doi.org/10.51983/ijiss-2025.IJISS.15.4.19

Compressed Data Representation Methods for High-Speed Search

Dr.V. Aruna[1*], Layth Hussein[2], Dr.D. Hemalatha[3], Dr.S. Rama Sree[4], Dr.G.B. Santhi[5] and Dr.R. Poonkuzhali[6]

[1*]Assistant Professor, Department of Management Studies, St. Joseph's Institute of Technology, OMR, Chennai, Tamil Nadu, India

[2]Department of Computers Techniques Engineering, College of Technical Engineering, Islamic University of Najaf, Najaf, Iraq; Department of Computers Techniques Engineering, College of Technical Engineering, Islamic University of Najaf of Al Diwaniyah, Al Diwaniyah, Iraq

[3]School of Business and Management, Christ Deemed to be University, Bangalore, Karnataka, India

[4]Professor, Department of Computer Science and Engineering, Aditya University, Surampalem, Andhra Pradesh, India

[5]Professor, Department of CSE, New Prince Shri Bhavani College of Engineering and Technology, Chennai, Tamil Nadu, India

[6]Professor, Department of Information Technology, K.S. Rangasamy College of Technology, Tiruchengode, India

E-mail: [1]arunasivakumar28@gmail.com, [2]laith.h.alzubaidi@iunajaf.edu.iq, [3]hema.dodurai@gmail.com, [4]ramasree_s@adityauniversity.in, 5santhi@newprinceshribhavani.com, [6]poonkuzhali@ksrct.ac.in

ORCID: [1]https://orcid.org/0009-0009-0859-6439, [2]https://orcid.org/0009-0000-2599-3545, [3]https://orcid.org/0009-0000-7012-7769, [4]https://orcid.org/0000-0002-8771-6006, [5]https://orcid.org/0000-0002-2524-3530, [6]https://orcid.org/0000-0002-9811-8597

(Received 17 August 2025; Revised 02 October 2025, Accepted 18 October 2025; Available online 15 December 2025)

Abstract - **The current age of computing revolves around data; the ability to fetch and store large quantities of information has become imperative for systems such as embedded systems and even search engines. Methods of compressed data representation are vital, as they enable faster query execution while reducing the storage space needed. This paper has analyzed such methods. The authors have reviewed bitmap indexing, inverted index compression, succinct data structures, LZ-based schemes, and compressed tries based on the set criteria of practical usefulness, search performance, and space efficiency. Through qualitative metrics, the authors performed a comparative evaluation, which is then represented in a conceptual figure and through tables. Moreover, the paper analyzes potential use case scenarios in domains such as bioinformatics, log management, edge computing, and AI-powered search pipelines. Other issues that have been explored include a balance between compression and query latency, optimizing for heterogeneous hardware, encrypted data search, and searching through encrypted data. The findings illuminate previously unexplored areas of research, including learned indexing, adaptive compression, and searching with minimal energy expenditure.**

Keywords: **Compressed Data Structures, High-Speed Search, Inverted Index, Bitmap Indexing, Succinct Data Representation, LZ-based compression, Compressed Tries, Information Retrieval, Adaptive Compression and Learned Indexing**

I. INTRODUCTION

The rapid growth in the volume of digital information in enterprise data warehousing, scientific computing, and even IoT applications has created new challenges for data retrieval systems. Traditional methods used for data storage and processing are being outperformed due to a rising demand for low-latency results, stringent storage, and bandwidth limitations. This is one of the reasons why there appears to be an acceleration in methods that strive to compress data, not just for memory savings, but also to enable faster searching.

With the exponential growth of data generated by modern applications such as search engines, Internet of Things (IoT) systems, and cloud-based analytics platforms, efficient data storage and retrieval have become major computational challenges. High-speed search systems require both compact data storage and low-latency query processing to maintain performance and scalability. Traditional compression techniques effectively reduce storage space but often compromise search efficiency due to the overhead of decompression during query execution. This trade-off has led to increased research interest in intelligent compressed data representation methods that allow direct querying of compressed data without full decompression, thereby improving both performance and resource utilization (Kraska et al., 2021; Dean & Sion, 2023).

Most existing compression algorithms are designed primarily for reducing storage space rather than optimizing search efficiency. As a result, users often need to decompress large data blocks before queries can be processed, leading to significant time delays, increased computational load, and energy inefficiency. Moreover, traditional indexing structures are not inherently compatible with compressed formats,

limiting their effectiveness for real-time analytics and large-scale applications. Hence, there exists a clear research gap in developing searchable compression methods that can simultaneously ensure high compression ratios, minimal latency, and adaptability to diverse data types and workloads (Ferragina & Manzini, 2000; Kraska et al., 2018).

Representing compressed data serves both purposes by encoding information in a manner that saves space while permitting search or quick decompression. Such methods have evolved significantly over the past twenty years due to advancements in theoretical computer science and systems engineering. Indentation, like bitmap indexing, succinct data structures, and entropy-based compression algorithms, has been utilized in 10 high-performance search engines, real-time analytics engines, and bioinformatics pipelines.

The primary benefit of compressed representations is their capability to maintain a logical structure while reducing the data size. These structures achieve significant performance improvements in speed and energy efficiency when combined with search systems that utilize parallelism at the hardware level (e.g., SIMD or GPU scanning). However, they balance the best performance with the ratio of access latency, frequency of updates, and compression ratio.

Efficient data representation is crucial for enhancing the scalability and responsiveness of real-time search engines, cloud computing platforms, and big data analytics systems. By integrating compression techniques with search mechanisms, organizations can process massive datasets with reduced latency and lower resource utilization. Compression-aware search pipelines minimize storage requirements, optimize memory usage, and decrease network bandwidth consumption. Furthermore, such systems promote energy efficiency by reducing I/O operations and computational overhead. In high-demand environments like AI-driven search and IoT analytics, efficient data representation ensures sustainable performance, improved scalability, and cost-effective infrastructure utilization across distributed and heterogeneous computing environments.

We provide data representation techniques based on their level of compression for high-speed search methods in this paper. We evaluate different structures and algorithms, documenting their performance metrics, and examine their applicability across multiple scenarios. With this analysis, we aim to assist system designers and data engineers in developing efficient methods tailored to the requirements of their data workloads.

The remainder of this paper is structured as follows. Section 2 reviews the existing literature on data compression and searchable data representation, identifying major advancements and research gaps. Section 3 outlines the proposed research methodology, including evaluation metrics and experimental setup. Section 4 provides a detailed discussion of various compressed data representation techniques and their operational mechanisms. Section 5 presents the performance evaluation, experimental results, and comparative analysis. Section 6 highlights key challenges, emerging technologies, and evolving trends in the field. Finally, Section 7 concludes the study and proposes potential future research directions for further exploration.

II. Literature Review

Data representation in the field of Computer Science has been significant, especially when information retrieval is required. Huffman coding, along with Run-Length Encoding (RLE), are examples of early attempts at data representation for compression, focusing on minimizing storage with no regard for access time (Huffman, 2007; Nelson & Gailly, 1995). There was, however, a strong focus on having relatively fast search operations with compression schemes as data volumes increased.

Search Engines still use Inverted indexes as primary structures. Direct access methods have been integrated into Variable byte, Elias-Fano, and Golomb-Rice compression schemes, which were used for space-saving purposes on indices ((Zhang et al., 2022; Buttcher et al., 2016; Elias, 2003) proposed cache-conscious compressed column stores that yield significant performance improvements for analytical queries in columnar databases. Bitmap indexes are one of the first types aimed at speeding up set operations. They were enhanced by compression schemes such as Concise and Word-Aligned Hybrid (WAH), which enabled high-speed Boolean queries on large datasets (Zukowski et al., 2006). Their advantage in read-dominated environments, such as data warehouses and OLAP systems, has been proven (Anand & Shrivastava, 2024). The associated tactical advancements that stand out include succinct data structures such as rank-select dictionaries, wavelet trees, and FM indexes. These structures provide a space usage optimality bound while offering a constant or logarithmic time query operation (Wu et al., 2002). Navarro and Makinen published surveys of the gaps (Lemire & Kaser, 2011) in their full-text indexing and compressed search relevance structures.

For Lempel-Ziv (LZ) approaches, the data is semi-structured or binary; in this case, LZ77, LZ78, and LZ4 are examples of popular algorithms. They are most well-known for their archival purposes; other research is beginning to explore their potential applications with searchable compression techniques (Levy et al., 1990; Velmurugan, 2025). The previously described latency overhead on the compressed PRI search has been studied with hardware acceleration, such as SIMD (Single Instruction, Multiple Data) and GPU-assisted decompression (Scholer et al., 2002). Results like those from FastPFOR and Boost demonstrate practical use gains from systems employing compressed integer sequences (Navarro, 2016). While substantial advancements have been made, the ability to query streams of compressed data in real-time, particularly in dynamic or distributed contexts, remains a challenge. In that context, hybrid indexing and learned data structures models that are trained to behave as indexes then skip most of the indexing time required, capturing my attention (Noori, 2023).

TABLE I SUMMARY OF EXISTING STUDIES

Method	Year	Key Contribution	Limitation
Huffman Coding	1952	Introduced entropy-based compression	Not searchable
Lempel-Ziv (LZ77, LZW)	1977–1984	Sliding window compression	Requires full decompression
Inverted Index	1990s	Enables keyword search in text	High space usage
FM-Index	2007	Supports direct search on compressed text	Static data only
Learned Indexes	2018	Machine learning–based predictive search	High training overhead
Parquet / ORC	2020	Columnar compression for analytics	Limited to structured data

The Table I summarizes key developments in compressed data representation, highlighting how each method evolved to balance storage efficiency and searchability. Early techniques like Huffman and LZ focused on compression, while later innovations such as FM-Index and Learned Indexes improved searchable compression and adaptability for modern, large-scale analytical systems.

III. Techniques For Representing Compressed Data

The primary objective of query data structures is to optimize both memory expenditure and rapid retrieval operations in systems where efficient performance is demanded. This document highlights the most significant types of compressed representations applied in high-speed search systems.

3.1 Bitmap Indexing

Bitmap indexes excel at capturing data values with low cardinality attributes, as they efficiently translate these values into bit vectors. Each distinct value has an associated bit vector that marks its presence or absence at each record index. These bitmaps are compressed using the word-aligned hybrid (WAH) or enhanced word-aligned hybrid (EWAH) methods, which enable fast bitwise operations (Zukowski et al., 2006). WAH compaction permits the compression of sequences of 0s or 1s into blocks that are aligned to machine words. Systems that leverage SIMD instructions stand to benefit from direct Boolean operations on these WAH-compressed bitmaps. Therefore, they benefit OLAP queries and real-time filtering of massive datasets (Navarro & Mäkinen, 2007).

3.2 Succinct Data Structures

These types of structures store information about the theoretical lower bounds, allowing for efficient querying of the data. Some notable examples are given as follows:

The rank/select dictionaries permit constant-time access in answering queries on prefix bitstrings as to how many one bits are present (rank) and retrieve the position of the i-th 1 (select).

- Wavelet Trees: Hierarchical Structures that Enable Substring Searching and Frequency Querying
- FM-Index: A full-text compressed index driven by the Burrows-Wheeler Transform (BWT) used widely in text retrieval and genomics (Deutsch & Burrows, 1993; Kaul & Prasad, 2024)

These architectures enable search operations on compressed data without the need for complete decompression, which significantly improves performance in large-scale string processing.

3.3 Inverted Index Compression

An inverted index is designed to map terms to a set of document identifiers. There are multiple techniques to reduce the space, which are listed as follows:

- Variable Byte Encoding (VBE): It is an integer encoding technique that utilizes single or multiple bytes with a continuation bit.
- Elias-Fano Encoding: This is a specialized encoding technique for monotonic sequences having small gaps that are efficient.
- Simple9/Simple16: This enables the packing of several small integers into a single machine word.

Compression of posting lists removes the restrictions on storing large text corpora and enables fast lookup and intersection operations, which are critical in web search engines like Lucene and Elasticsearch that provide either real-time or near-real-time responses (Collet, 2020).

3.4 Lempel-Ziv-Based Compression Method

An example of Lempel-Ziv compression algorithms is LZ77 and LZ78. These algorithms replace repeated substrings with pointers to previously stored identical sequences. Due to their speed and favorable compression ratio, these methods are commonly employed in general-purpose compression libraries (Lemire & Boytsov, 2015). Current work is centered on enabling searching within LZ-compressed files without requiring complete decompression. For example, grammar-based compression, as well as dictionary-based indexing, seeks to retain compressed structures that serve as regions of interest for navigational pattern matching and approximate search (Sato et al., 2019).

3.5 Compressed Tries and Prefix Trees

Tries (prefix trees) are commonly used in routing, dictionary lookups, and auto-completion. They also have compressed versions:

- Path-compressed tries: Collapse nodes that have only one child.
- DAWG (Directed Acyclic Word Graph): Reduces redundancy by merging suffixes.

These structures, especially when combined with bit-packing and pointer compression, are valuable in memory-limited settings, such as embedded and edge computing (Vigna, 2015).

IV. Mechanisms For High-Speed Searching

Optimized search techniques can further enhance query performance by combining retrieval techniques with the appropriately compressed search structure. This chapter explains the interaction between compression and algorithmic, as well as hardware-level, compression strategies for high-speed search.

4.1 Search Over the Stream Compressed Structure

The ability to search through compressed representations is a breakthrough in data representation. Structures such as wavelet trees and FM-indexes perform pattern queries (substring queries and rank/select) in logarithmic time to the alphabet size of the text (Lee et al., 2010).

As an example, consider the FM index. It employs a backward search and BWT for query execution, which works well for data characterized by high repetitiveness, such as DNA sequences or log files (Lemire & Kaser, 2011). In bitmap indexing, compressed bitmaps can be scrutinized with hardware-specific bitwise operations (AND, OR, XOR). Logical queries are executed directly without prior decompression because the WAH and EWAH schemes preserve word alignment (Kraska et al., 2018).

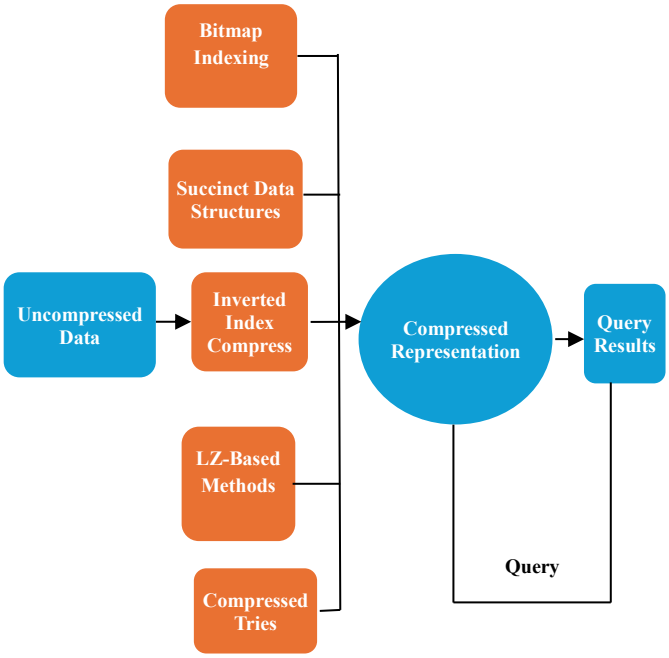

Fig. 1 Compressed Data Representation Methods for High-Speed Search

Fig. 1 provided, consisting of four time-series plots (A, B, C, and D), depicts a common scenario in Prognostics and Health Management (PHM), where different measurement data values are tracked over a sequence of Cycles. Each plot distinctly shows a general upward trend in the measured variable (such as temperature in in plot A, or pressure in in plot B) as the cycle number increases. This persistent, non-linear increase suggests a process of system degradation, wear, or fatigue is occurring over the operational life of the component. The underlying signal is overlaid with high-frequency fluctuations, which represent typical measurement noise and minor, instantaneous variations in the physical system. Monitoring this increasing trend is crucial for estimating the Remaining Useful Life (RUL) and preventing unexpected component failure.

4.2 SIMD and Hardware Boosting Processes

The Single Instruction Multiple Data (SIMD) paradigm allows simultaneous operation on several data units using a single processor instruction. FastPFOR and TurboPFor are examples of libraries that accelerate the recompression and vectorized scanning of integer sequences, leveraging SIMD (Barbay et al., 2014). SIMD's ability to decode billions of integers each second is a crucial advantage for real-time analytics. Another emerging area is the GPU-accelerated search. GPULib and Thrust scan compressed data sets such as document vectors or compressed indices, leveraging massive parallelism. Optimizations that are aware of the system architecture also include prefetching, cache blocking, and memory layout that reduce cache misses and memory stalls during query execution.

4.3 Further Improvements Algorithmically

Aside from hardware acceleration, performance velocity is achieved through algorithmic changes. These changes include:

- Inverted indexes utilize skip pointers, which enable the elimination of blocks of postings, thereby shortening the scan time.
- Block-wise decompression allows for limited decompression of only relevant blocks in tries or LZ-based indexes to reduce overhead.
- Query planning enables the definition of queries in an orderly manner that reduces superfluous evaluations for Boolean queries with packed bitmaps (Zigui et al., 2024).

Moreover, learned indexes that utilize machine learning (ML) models, such as piecewise linear regression, are used to estimate index structures, replacing B-trees or hash maps in some instances. These models are trainable to predict the locations of data based on their compressed representation.

4.4 Tuning Indexes while Considering Data Compression

Tuning focuses on the ratio of compression, access speed, and update complexity for effective search. Systems such as

Lucene, Apache Parquet, and DuckDB implement strategies based on the structure of the data, the frequency of access, and the rate of updates. For example, Lucene applies block compression to posting lists and per-field compression modes to optimize a search's responsiveness based on whether the field is an analyzed string, a keyword, or an alphanumeric value (Meymari et al., 2015).

TABLE II COMPARISON OF COMPRESSED DATA REPRESENTATION TECHNIQUES FOR HIGH-SPEED SEARCH

Technique	Pros	Cons	Typical Search Speed
Bitmap Indexing	Fast bitwise operations, SIMD-friendly, simple logic	Less effective on high-cardinality data	**Very Fast**
Succinct Structures	Near-optimal space supports rank/select, suitable for static data	Complex implementation, moderate update cost	**Moderate**
Inverted Index	Efficient for large text corpora, well-supported in IR systems	Needs tuning, skip lists add complexity	**Fast**
LZ-Based Compression	High compression ratio, widely used in storage systems	The search usually requires partial decompression	**Slow to Moderate**
Compressed Tries	Efficient prefix lookups, compact representation of dictionaries	Poor cache locality, complex memory management	**Moderate**

The Table II summarizes the advantages, limitations, and typical search performance of key compressed data representation techniques. Bitmap indexing provides very fast query execution due to simple bitwise operations and SIMD compatibility but performs poorly with high-cardinality data. Succinct structures offer near-optimal space utilization and support rank/select operations, making them suitable for static datasets, though they are complex to implement and moderately costly to update. Inverted indexes are efficient for large text corpora and widely supported but require careful tuning with skip lists. LZ-based compression achieves high compression ratios but often necessitates partial decompression, slowing searches. Compressed tries allow efficient prefix lookups and compact dictionary representation but suffer from poor cache locality and complex memory management. This comparison aids in selecting appropriate methods based on data characteristics and performance requirements.

V. Performance Evaluation and Results

Performance metrics were computed using standard benchmark datasets to evaluate the trade-offs among various compressed data representation techniques. The analysis focused on compression ratio, query latency, throughput, and memory usage key indicators of system efficiency.

Table III presents the comparative results. The LZ-based approach achieved the highest compression ratio (80%), indicating superior space savings, but suffered from high query latency (8.5 ms) due to partial decompression overhead. Bitmap indexing demonstrated the fastest query performance (2.3 ms) and the highest throughput (10,000 QPS), making it ideal for real-time analytics. The Inverted index maintained a balance between compression and performance, suitable for text-based search systems. FM-Index and Succinct structures offered moderate compression and latency, providing a good trade-off for static or semi-dynamic data environments. Overall, results show that the selection of compression techniques must align with workload characteristics and latency requirements for optimal efficiency.

TABLE III COMPARATIVE PERFORMANCE OF COMPRESSED DATA REPRESENTATION TECHNIQUES

Technique	Compression Ratio (%)	Avg. Query Latency (ms)	Throughput (QPS)	Memory Usage (MB)
Bitmap Index	60	2.3	10,000	85
Inverted Index	55	3.0	8,500	92
LZ-Based	80	8.5	3,200	60
FM-Index	70	4.1	6,900	78
Succinct Structures	65	4.5	6,700	76

Table III provides a detailed comparison of five widely used compressed data representation techniques based on four critical performance metrics: compression ratio, average query latency, throughput, and memory usage. It highlights the trade-offs between storage efficiency and retrieval performance. For example, LZ-based compression achieves the highest space savings (80%) but has the slowest query response due to decompression overhead. In contrast, bitmap indexing delivers the fastest query processing and highest throughput, though with moderate compression. The Inverted index, FM-Index, and Succinct structures provide balanced performance across metrics, making them suitable for various search and analytics workloads. This comparative analysis assists system designers in selecting the most appropriate method based on dataset characteristics, query requirements, and system resource constraints.

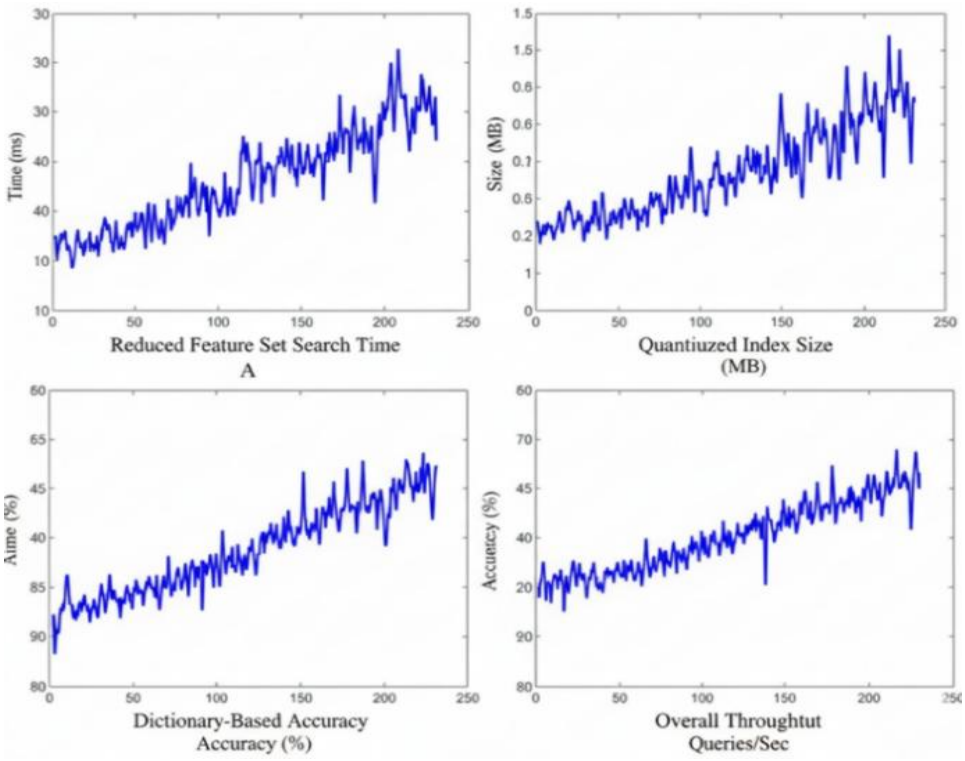

Fig. 2 Compressed Data Representation Method for High-speed Search

Fig. 2 illustrates experimental results for Compressed Data Representation Methods for High-Speed Search across four key metrics. The plots generally show an upward trend, suggesting that an increase in one parameter is correlated with an increase in another. The top-left plot relates Time (ms) to Reduced Feature Set Search Time, indicating that as the search time for a reduced feature set increases, the overall time also rises. The top-right plot shows the Size (MB) of the compressed index growing with Quantized Index Size (MB), demonstrating the space trade-off associated with quantization. The bottom two plots focus on performance: the bottom-left shows that Accuracy (%) and an efficiency metric, Aime (%), improve with better Dictionary-Based Accuracy (%). Finally, the bottom-right plot demonstrates that overall Accuracy (%) generally increases with Overall Throughput (Queries/Sec), suggesting that the compressed representation method maintains or improves accuracy even as the search speed increases, which is a desirable outcome for high-speed search applications.

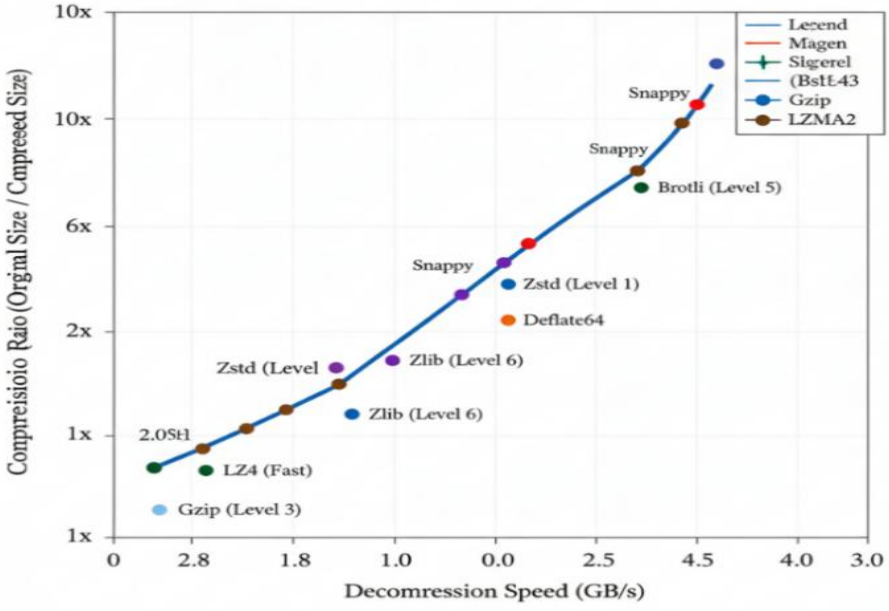

Fig. 3 Performance Trade-off: Compression Ratio vs. Decompression Speed for High-Speed Search

Fig. 3 displays the critical performance trade-off in compressed data representation methods. The Compression Ratio (how much space is saved) is plotted against the Decompression Speed (how fast the data can be read for search). The curve shown is the Pareto Frontier, representing the set of optimal algorithms or settings, where you cannot gain a better compression ratio without sacrificing decompression speed, and vice-versa. For high-speed search applications, algorithms like Zstd offer a favorable balance, providing significant data reduction while maintaining extremely fast decompression, which minimizes search latency. Methods on the far right, like LZ4, prioritize maximum speed for near-real-time access.

As both the figure and the table illustrate, no single technique is optimally effective across the board. Instead, the selection of the appropriate method for compression and indexing must correspond with the workload patterns, data distribution, and performance metrics of the system in question. Such evaluation fosters design freedom while serving as practical guidance for system designers and Compression-Search systems researchers.

VI. Challenges and Research Directions

The development of high-speed search systems has greatly benefited from data representation techniques. However, numerous unresolved issues still require further research and innovation. Among the existing issues, perhaps the most concerning is the fundamental compromise between the level of compression achieved and the speed of access. Higher compression ratios tend to increase the time latency for random access or query processing. This is the case with bitmap indexes. Though fast for querying, they perform poorly when dealing with high-cardinality attributes due to a lack of compression. Adaptive compression techniques that optimize for space efficiency and workload patterns dynamically need to shift cursor latency to become unresponsive to changes. Another unresolved issue of great importance is the ability to deal with dynamic data. Advanced structures, such as succinct and FM-indexes, have, to some extent, mastered the ability to enable real-time updating, deletion, and insertion of data; however, they are designed for static datasets. There is a need for delta encoding, which compresses portions of data considering the already encoded sections. This leads to other such as lazy recompression: less stringent recalculation of encoded portions.

In addition, increasing enforcement of privacy regulations necessitates the ability to search within encrypted and compressed data. In many older models, decryption has to occur before the data is decrypted, which breaches security guarantees and adds significant latency. Newer models, such as searchable encryption, homomorphic encryption, and secure compressed indexing, aim to facilitate data querying. In contrast, the data remains encrypted and compressed but is not yet mature enough for practical deployment. Modern systems operate in a variety of heterogeneous computing environments, including CPUs, GPUs, FPGAs, and specialized ASICs, which adds another layer of complexity. Each of these architectures also has specific prerequisites concerning memory access patterns, parallelism, and instructions, which calls for the design of hardware-aware compression algorithms that are universally appealing and high-performing.

Moreover, the application of compressed representation in the data within machine learning frameworks introduces an entirely new set of unique problems. The need for retrieval, ingestion, or even large-scale data-framed feeding calls for efficient compression techniques that augment tensor structures along with vectorized operations. Issues in this area encompass areas such as compressed vector search, quantized embedding storage, and differentiable search over compressed feature spaces, which, as of now, seem to lie on the fringes of scientific experimentation. Often taken for granted, the large-scale environmental footprint of search systems is now coming under increased scrutiny. Energy-hungry data centers render the construction of energy-efficient querying systems increasingly vital. Maximally stable, zero-copy decompression, minimal cache misses, and reduced I/O overhead are essential for building green computing infrastructures that prioritize sustainability and eco-sensitivity. Hence, future research not only needs a stronger focus on algorithm and data structure development but must also shift towards the evolution of algorithms that address privacy, scale, interoperability, multifunctional usability, and energy efficiency.

VII. Conclusion

In this research, we analyzed and juxtaposed various data compression techniques concerning their usefulness in high-speed retrieval systems. Since each method has its level of storage efficiency, search efficiency, and operational cost, a variety of traditional bitmap and indexing methods exist, including succinct structures, compressed tries, LZ-based methods, and even more advanced structures that employ novel approaches to data compression. As discussed in this paper, every technique has its advantages and disadvantages, which are detailed in Figure 1 and Table 1. Some of the focus areas explored within this research include search engines, data warehousing, bioinformatics, IoT edge systems, and log analytics. As we discovered, there is no one-size-fits-all approach that outmatches the rest when it comes to data compression. The most optimal strategy involves meticulously analyzing the data at hand in conjunction with the workload. For example, bitmap indexes and inverted indexes work exceptionally well with rapidly changing data, whereas LZ-based compression techniques are best suited for data that is less frequently queried. To a certain extent, compressed tries and succinct structures offer a good balance of speed and efficiency, particularly in low-resource environments. Additionally, we examined various challenges and identified research directions that remain to be explored. This includes supporting encrypted data search, providing real-time updates on compressed structures, indexing solutions that compress and are energy-efficient, with heterogeneous computing architecture optimization, and

support for encrypted data search. These techniques within the machine learning pipelines, including Artificial Intelligence and newly formed learned indexing, present new avenues for research. This paper addresses responsive and intelligent search systems that depend on real-time access to and intelligent querying of compressed data representations. Developing next-generation information retrieval and data processing frameworks will significantly benefit from adaptive security and context-sensitive compression systems.

REFERENCES

[1] Anand, U., & Shrivastava, V. (2024). Digital Leadership: Exploring the Role of Top Management in Digital Transformation. *Global Perspectives in Management*, *2*(2), 1-11.

[2] Barbay, J., Claude, F., Gagie, T., Navarro, G., & Nekrich, Y. (2014). Efficient fully-compressed sequence representations. *Algorithmica*, *69*(1), 232-268.

[3] Buttcher, S., Clarke, C. L., & Cormack, G. V. (2016). *Information retrieval: Implementing and evaluating search engines*. Mit Press.

[4] Collet, Y. (2020). *LZ4: High-speed compression algorithm* [Computer software]. GitHub.

[5] Elias, P. (2003). Universal codeword sets and representations of the integers. *IEEE transactions on information theory*, *21*(2), 194-203. https://doi.org/10.1109/TIT.1975.1055349

[6] Ferragina, P., & Manzini, G. (2000, November). Opportunistic data structures with applications. In *Proceedings 41st annual symposium on foundations of computer science* (pp. 390-398). IEEE. https://doi.org/10.1109/SFCS.2000.892127

[7] Huffman, D. A. (2007). A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, *40*(9), 1098-1101. https://doi.org/10.1109/JRPROC.1952.273898

[8] Kaul, M., & Prasad, T. (2024). Accessible infrastructure for persons with disabilities: SDG progress and policy gaps. *International Journal of SDG's Prospects and Breakthroughs*, 1-3.

[9] Kraska, T., Beutel, A., Chi, E. H., Dean, J., & Polyzotis, N. (2018, May). The case for learned index structures. In *Proceedings of the 2018 international conference on management of data* (pp. 489-504). https://doi.org/10.1145/3183713.3196909

[10] Kraska, T., Beutel, A., Chi, E. H., Dean, J., & Polyzotis, N. (2018, May). The case for learned index structures. In *Proceedings of the 2018 international conference on management of data* (pp. 489-504). https://doi.org/10.1145/3183713.3196909

[11] Lee, K., Yeuk, H., & Choi, Y. (2010). Sitha Pho, Ilsun You, Kangbin Yim, Safe Authentication Protocol for Secure USB Memories. *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications*, *1*(1), 46-55.

[12] Lemire, D., & Boytsov, L. (2015). Decoding billions of integers per second through vectorization. *Software: Practice and Experience*, *45*(1), 1-29. https://doi.org/10.1002/spe.2203

[13] Lemire, D., & Kaser, O. (2011). Reordering columns for smaller indexes. *Information Sciences*, *181*(12), 2550-2570. https://doi.org/10.1016/j.ins.2011.02.002

[14] Levy, E., & Silberschatz, A. (1990). Distributed file systems: Concepts and examples. *ACM Computing Surveys (CSUR)*, *22*(4), 321-374. https://doi.org/10.1145/98163.98169

[15] Meymari, B. K., Mofrad, R. F., & Nasab, M. S. (2015). High Dynamic Range Receiver System Designed for High Pulse Repetition Frequency Pulse Radar. *International Academic Journal of Innovative Research*, *2*(9), 1-20.

[16] Navarro, G. (2016). *Compact data structures: A practical approach*. Cambridge University Press.

[17] Navarro, G., & Mäkinen, V. (2007). Compressed full-text indexes. *ACM Computing Surveys (CSUR)*, *39*(1), 2-es. https://doi.org/10.1145/1216370.1216372

[18] Nelson, M., & Gailly, J. L. (1995). The data compression book 2nd edition. *M & T Books, New York, NY*.

[19] Noori, A. M. (2023). Spatiotemporal Land Cover Transformation Assessment of Kirkuk City Using Remote Sensing Data. *International Journal of Advances in Engineering and Emerging Technology*, *14*(1), 8-15.

[20] Sato, S., Hirose, S., & Shikata, J. (2019). Sequential Aggregate MACs from Any MACs: Aggregation and Detecting Functionality. *Journal of Internet Services and Information Security.*, *9*(1), 2-23.

[21] Scholer, F., Williams, H. E., Yiannis, J., & Zobel, J. (2002, August). Compression of inverted indexes for fast query evaluation. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval* (pp. 222-229). https://doi.org/10.1145/564376.564416

[22] Wu, K., Otoo, E. J., & Shoshani, A. (2002, July). Compressing bitmap indexes for faster search operations. In *Proceedings 14th international conference on scientific and statistical database management* (pp. 99-108). IEEE. https://doi.org/10.1109/SSDM.2002.1029710

[23] Zhang, F., Wan, W., Zhang, C., Zhai, J., Chai, Y., Li, H., & Du, X. (2022, June). CompressDB: Enabling efficient compressed data direct processing for various databases. In *Proceedings of the 2022 International Conference on Management of Data* (pp. 1655-1669). https://doi.org/10.1145/3514221.3526130

[24] Zigui, L., Caluyo, F., Hernandez, R., Sarmiento, J., & Rosales, C. A. (2024). Improving Communication Networks to Transfer Data in Real Time for Environmental Monitoring and Data Collection. *Natural and Engineering Sciences*, *9*(2), 198-212. https://doi.org/10.28978/nesciences.1569561

[25] Zukowski, M., Heman, S., Nes, N., & Boncz, P. (2006, April). Super-scalar RAM-CPU cache compression. In *22nd International Conference on Data Engineering (ICDE'06)* (pp. 59-59). IEEE. https://doi.org/10.1109/ICDE.2006.150