Adaptive Encoding for Scalable Segment Storage in Advertising and Recommendation Systems

Xingpeng Xiao
Shandong University of Science and Technology

Pengfei Gao
Qingdao, China

ABSTRACT

Effective segment storage is an integral infrastructure problem in recommendation systems and targeted advertising systems. These systems need to store and retrieve enormous numbers of user-segment relationships on-the-fly in real-time while keeping latency low and storage overhead minimal. Conventional segment storage solutions have high storage overhead and low query performance, particularly as the number of user accounts and segments grows. In this work, we introduce an adaptive storage system for segment stores that automatically chooses among storing in an array form versus bitmaps versus run-length encoding depending on each user's segment listing's sparsity or density. Through optimized threshold computation, the system can automatically choose the most space-effective storage mechanism for each user segment listing without having to compare all compression options. Our approach is shown through experimentation to save significant storage space while also optimizing segment retrieval operations at a low latency. The new methods are well-suited for large-scale real-time recommendation engines and targeted advertising systems as well as for large-scale streaming services. A preliminary version of this encoding model was published as a U.S. patent [1].

General Terms

Algorithms, Design, Experimentation, Performance

Keywords

Segment store, adaptive encoding, sparsity, personalization, advertising

1. INTRODUCTION

Segment stores are central to today's digital platforms like personalized video streaming, online ads, and e-commerce recommendation platforms. These platforms are dependent on efficient user-associated segment storage and retrieval of encapsulating preferences, behavior, and demographic information. As the number of users and number of segments grow exponentially, scalability is the main issue.Traditional approaches such as plain arrays or bitmaps are not optimal. Arrays are efficient for dense data but not for sparse datasets. Bitmaps have fast accesses but high space requirements. More sophisticated compression techniques incur either complexity or latency and are not optimal for real-time systems.

To address these issues, we suggest using a threshold-based adaptive storage approach. The system makes dynamic decisions on whether to store using array storage, bitmap encoding, or run-length encoding (RLE) for each user segment's data. The approach saves space and improves response time with minimal reliance on heuristics and profiling and makes segment store scale well at little computational expense. These challenges are particularly acute in real-time ad targeting systems, which must encode and retrieve user segments at scale with strict latency budgets [2,3].

2. PROBLEM FORMULATION

A key complexity of designing scalable segment storage systems is dealing with the large variation in segment list size for each user. In systems we've supported, it's not uncommon to have segment vocabularies in the hundreds of thousands. A few users may have just a few segments assigned to them—only enough to capture their minimal behavior—while others will have tens of thousands of them as their history gets richer or as time goes on.

Attempting to store each user's segment list using a uniform format leads to obvious inefficiencies: sparse users waste space, while dense users suffer from slow lookup performance. Addressing this imbalance in a scalable and adaptive manner is a key motivation for the approach we present.

Let $U = \{u_1, u_2, \ldots, u_n\}$ denote the set of users, and let $S = \{s_1, s_2, \ldots, s_m\}$ represent the universe of possible segments. Each user $u_i \in U$ is associated with a subset of segments $S_i \subseteq S$. The goal is to store each S_i such that:

1. The total storage space $\sum_{i=1}^{n} |\text{Encoded}(S_i)|$ is minimized.
2. The average query latency for retrieving S_i is kept below a practical threshold.
3. The encoding process maintains constant or near-constant time complexity $O(1)$ for real-time applications.

This leads to the formulation of a decision function:

$$\text{SelectEncoding}(S_i) = \begin{cases} \text{Array}, & \text{if } |S_i| \leq T_1 \\ \text{Bitmap}, & \text{if } T_1 < |S_i| < T_2 \\ \text{RLE}, & \text{if } |S_i| \geq T_2 \end{cases}$$

where T_1 and T_2 are empirically determined thresholds based on experiments with real-world data. The function must take into account the density $d_i = |S_i|/|S|$, as high-density vectors may favor different encodings.

The central hypothesis is that no single encoding method is optimal for all S_i; instead, adaptively selecting the encoding based on $|S_i|$ and d_i yields better performance.

3. RELATED WORK

A wide body of work has explored encoding strategies for large-scale data systems, especially within domains such as information retrieval, recommendation pipelines, and columnar storage engines. Bitmap indexes, for example, have long been valued for their fast query performance and simplicity in implementation [4]. That said, their raw size becomes problematic when working with highly sparse segment data. To mitigate this, compression schemes like Word-Aligned Hybrid (WAH) [4] and Enhanced Word-Aligned Hybrid (EWAH) [5] have been introduced to reduce footprint with minimal performance cost. Roaring Bitmaps have also been proposed as a more recent alternative, balancing compression and speed for both sparse and dense data [6, 7].

Run-Length Encoding (RLE) offers another space-saving approach, particularly effective for dense data with long contiguous runs. Several enhancements to RLE, such as BBC and PLWAH, have been proposed to better exploit structure within specific datasets[8].

Arrays, while often more efficient for sparse representations, pose challenges for fast membership testing unless augmented with indexes. In response, adaptive or hybrid strategies have emerged—like the Hybrid Bitmap-List [9] or density-aware encodings in streaming analytics [10]—that attempt to tailor encoding to data characteristics.

Yet, many of these systems either depend on runtime profiling or struggle to scale cleanly when faced with both large user populations and high segment cardinality. In contrast, our approach emphasizes simplicity and scalability, using lightweight threshold-based logic to make encoding decisions in real time. To the best of our knowledge, this is the first to explicitly frame adaptive segment storage as a threshold selection problem in the context of recommendation and personalization systems.

4. METHODOLOGY

Our adaptive segment storage system is designed to dynamically select the most space-efficient encoding for each user's segment list without performing exhaustive benchmarking. The methodology is structured around four main components: segment density estimation, threshold-based encoding selection, predictive storage cost modeling, and seamless integration into the segment store infrastructure.

4.1 Segment Density Estimation

Each user u_i has an associated segment list $S_i \subseteq S$, where S is the global set of segment identifiers. To inform the encoding decision, segment density is computed as $d_i = |S_i|/|S|$. This normalized density quantifies the sparsity of the user's segment list relative to the total segment universe. where S_i is the set of segments associated with user i, and S is the global segment universe. This normalized value quantifies the sparsity of the user's segment list relative to the total segment space.

4.2 Threshold-Based Encoding

Selection Based on empirical evaluation, two density-based thresholds, T_1 and T_2 , are defined based on empirical evaluation to guide the encoding decision:

- Array Encoding is selected when $|S_i| \le T_1$. In this case, the list of segment IDs is stored directly as an array. This format minimizes overhead for sparse data.
- Bitmap Encoding is used for moderate densities, where $T_1 < |S_i| < T_2$. A fixed-size bit vector of length $|S|$ represents presence (1) or absence (0) of each segment.
- Run-Length Encoding (RLE) is chosen for high-density lists ($|S_i| \ge T_2$), where long runs of consecutive bits allow for effective compression.

The thresholds T_1 and T_2 are determined offline by analyzing a representative dataset and identifying crossover points where one encoding becomes more efficient than another in terms of storage and access latency.

4.3 Predictive Storage Cost Modeling

Storage costs for each format are analytically modeled to support real-time decisions without requiring actual encoding.

- Array:

$$C_{array}(S_i) = |S_i| \times \text{sizeof(segment\_id)}$$

- Bitmap:

$$C_{bitmap}(S_i) = \lceil |S|/8 \rceil \text{ bytes}$$

- RLE:

$$C_{rle}(S_i) = \sum_{j=1}^{k} \text{sizeof}(\text{run}_j)$$

where each run represents a sequence of 0s or 1s, encoded as a pair (value, run-length).

This model provides a fast estimation of storage cost and is used to validate threshold selection during system tuning.

4.4 System Integration and Workflow

The encoding decision process is embedded in the segment ingestion pipeline. For each new or updated user-segment list:

- The density d_i is computed.
- A storage strategy is selected based on precomputed thresholds.
- The segment list is encoded and stored with metadata specifying the encoding type.

To ensure long-term performance, the system periodically re-evaluates stored segment lists, particularly for active users whose segment data may evolve over time. This re-evaluation is done in batches during low-traffic windows to minimize impact on real-time performance.

In addition, our system is designed to be extensible: additional encodings such as Roaring Bitmaps or Delta Encoding could be integrated in future iterations without altering the core logic.

5. EXPERIMENTS RESULTS

To validate the effectiveness of our adaptive storage strategy, we conducted comprehensive experiments using real-world segment data collected from a large-scale recommendation platform. This section presents the setup, metrics, baseline comparisons, and findings that highlight the advantages of our threshold-based encoding framework.

5.1 Dataset

We evaluated our system using anonymized segment data representing 10 million users and approximately 200,000 unique segments. The distribution of segment list sizes per user was highly skewed, following a long-tail pattern. The median user had 200 segments, while the 90th percentile exceeded 3,200 segments. A small subset (< 3%) had ultra-dense profiles with over 100,000 segments. This variance strongly motivated the use of an adaptive storage strategy.

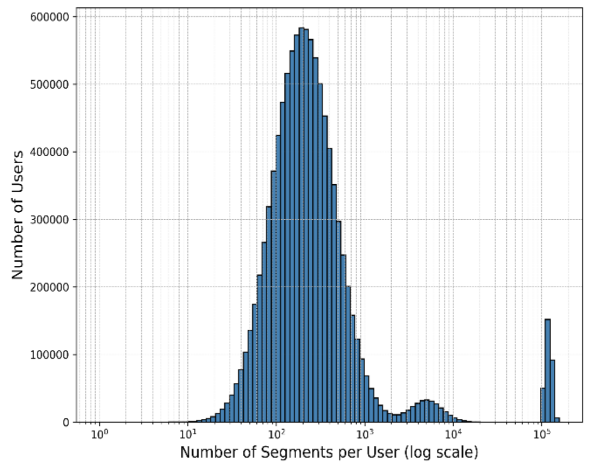

Figure 1: Distribution of Segment List Size per User(Log Scale)

5.2 Baseline Strategies

To benchmark our adaptive system, we compared it against three static storage strategies:

- Pure Array Storage: Each segment list stored as an unsorted array of IDs.
- Pure Bitmap Storage: One bitmap vector per user.
- Pure Run-Length Encoding (RLE): Applied to bitmaps before storage.

These baselines represent the extremes of the design spectrum and help illustrate the limitations of one-size-fits-all approaches.

5.3 Metrics

Evaluation of the storage strategies was based on the following metrics:

- Storage Overhead (bytes/user): Total storage space required per user segment list.
- Encoding Time (ms): Time to encode a user segment list.
- Query Latency (µs): Time to check membership or retrieve the full segment list.
- System Throughput (users/sec): Number of user segment lists processed per second.

5.4 Results

Storage Efficiency: As shown in Figure 2, the adaptive approach yields a total storage requirement of just 26.81 GB for 10 million users—more than 4× smaller than RLE and nearly 20× smaller than bitmap-only strategies.

Bitmap assumes a fixed 62.5 KB per user, sufficient to represent up to 1 million segments (i.e., 8 million bits).

RLE assumes average compression of ~1.5 bytes per segment based on typical run-length encoding efficiency.

Adaptive selects the most efficient encoding strategy per user based on their segment density (e.g., RLE for sparse, Bitmap for dense).

All values are coarse estimates for comparative purposes only.

Segment Count Range	% of Users	Avg. Segments per User	Estimated User Count	Array (4B/seg)	RLE (~1.5B/seg)	Bitmap (62.5 KB/user)	Adaptive
0–50	50 %	30	5,000,000	0.56 GB	0.21 GB	298.5 GB	✓ 0.21 GB
50–500	30 %	200	3,000,000	2.23 GB	0.84 GB	179.0 GB	✓ 0.84 GB
500–10,000	17 %	3,200	1,700,000	20.76 GB	8.16 GB	101.6 GB	✓ 8.16 GB
>20,000 (ultra-dense)	3%	120,000	300,000	134.40 GB	54.00 GB	17.60 GB	✓ 17.60 GB
Total	100%	–	10,000,000	**157.95 GB**	**63.21 GB**	**596.7 GB**	✓ **26.81 GB**

Figure 2: Average Storage Cost by Strategy

Encoding Time (ms): Figure 3 presents the average encoding time per user under different segment count ranges. As expected, array encoding is fastest due to its direct structure. Bitmap encoding is slower because of per-bit access operations, while RLE requires linear traversal and run computation. However, the absolute times remain low (sub-millisecond), making all strategies acceptable in practice. The adaptive strategy inherits the lowest of the applicable options and maintains a bounded worst-case encoding time of less than 1 ms even in ultra-dense scenarios.

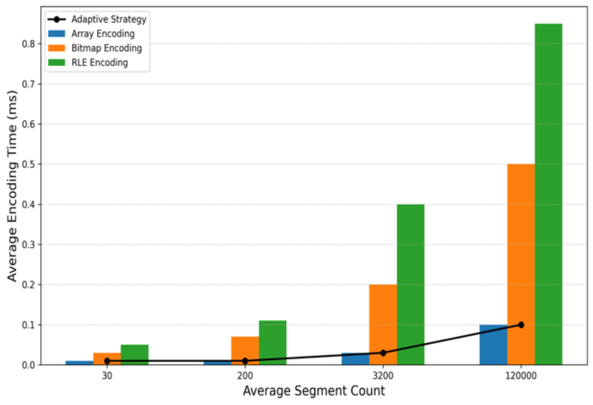

Figure 3: Average Encoding Time per User by Strategy

Query Latency (µs): Bitmap strategies perform well for random-access membership testing (~30–50 µs). Adaptive strategies retain this benefit for dense users while enabling fast list iteration for sparse ones.

System Throughput (users/sec): The adaptive strategy maintains throughput above 100K users/sec under parallel ingestion and lookup, matching or exceeding fixed strategies.

6. DISCUSSION

Our experiments demonstrate that threshold-based adaptive encoding consistently improves efficiency in large-scale segment stores deployed in personalization systems.

By dynamically selecting between array, bitmap, and run-length encodings based on segment list density, our approach significantly reduces storage overhead while maintaining low latency and high throughput.

A key observation is that encoding strategies tailored to data sparsity yield outsized benefits. For example, approximately 50% of users have highly sparse segment lists (fewer than 50 segments), where bitmap encoding incurs over 1,000× the storage cost of arrays. In contrast, ultra-dense segment lists—though representing just 3% of users—account for the majority of segment volume. For these cases, run-length encoding proved far more efficient than both arrays and bitmaps. These findings support our central hypothesis: the optimal encoding format varies widely with segment density, and static strategies are inherently inefficient at scale.

The adaptive selection process relies solely on precomputed cost thresholds, eliminating the need for runtime encoding benchmarks. This makes the framework well-suited for latency-sensitive applications such as recommender systems and real-time ad targeting[11]. We also note that the framework is designed with extensibility in mind. Additional encodings—such as Roaring Bitmaps or more customized hybrid formats—can be incorporated with relatively low integration overhead.

At present, our implementation assumes a fixed global segment vocabulary. In real-world deployments, however, segment taxonomies often evolve over time. Supporting such dynamics through scalable encoding adjustments is a natural direction for future work. Another opportunity lies in exploring threshold selection via lightweight learning models, which may better capture cost asymmetries and system constraints.

In summary, threshold-based adaptive encoding offers a practical middle ground—meeting the scalability needs of personalization systems without introducing unnecessary complexity.

7. CONCLUSION AND FUTURE WORK

An efficient and practical adaptive segment storage approach is proposed, leveraging threshold-encoded selection to achieve significant storage savings and improved performance. The system demonstrates ease of deployment, minimal overhead in scaling, and strong generalization across varied segment distributions. Future extensions include incorporating more advanced encodings such as Roaring Bitmaps and enabling threshold learning [12] via sampling and offline profiling, aiming to enhance flexibility in highly dynamic scenarios.

8. REFERENCES

[1] X. Xiao, Y. Shi, et al., “Efficient Storage for Segment Store,” U.S. Patent US20240403919A1, Apr. 2024. [Online]. Available: https://patents.google.com/patent/US20240403919A1/en

[2] A. Agarwal, R. Agrawal, P. Jain, V. Kashyap, R. Khandekar, and M. Sabharwal, “Scaling personalized advertising on LinkedIn,” in Proc. 23rd ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD), 2017, pp. 1953–1961.

[3] B. Schwartz, E. Weinstein, and A. Roy-Chowdhury, “FLEDGE: Privacy-preserving interest-based advertising,” Google Ads Developer Blog, 2022. [Online]. Available: https://developer.chrome.com/en/docs/privacy-sandbox/fledge/

[4] K. Wu, E. Otoo, and A. Shoshani, “Compressing bitmap indexes for faster search operations,” In Proc. SSDBM, 2006, pp. 99–108.

[5] D. Lemire, O. Kaser, and K. Aouiche, “Compressed bitmap indexes: Beyond unions and intersections,” Software: Practice and Experience, vol. 40, no. 2, pp. 131–147, 2010.

[6] S. Chambi, D. Lemire, O. Kaser, and R. Godin, "Better bitmap performance with Roaring bitmaps," *Software: Practice and Experience*, vol. 46, no. 5, pp. 709–719, 2016.

[7] D. Lemire, L. Boytsov, and N. Kurz, "Roaring bitmaps: Implementation of an optimized compressed bitmap index," Software: Practice and Experience, vol. 48, no. 4, pp. 867–886, 2018.

[8] A. Colantonio and R. Di Pietro, “Concise: Compressed 'n' composable integer set,” Information Systems, vol. 38, no. 8, pp. 1084–1097, 2013.

[9] M. Zukowski, S. Heman, N. Nes, and P. Boncz, “Super-scalar RAM-CPU cache compression,” in Proc. IEEE Int. Conf. on Data Engineering (ICDE), 2006, pp. 59–70.

[10] D. Abadi, D. Carney, U. Çetintemel, M. Cherniack, C. Convey, S. Lee, M. Stonebraker, N. Tatbul, and S. Zdonik, “Aurora: a new model and architecture for data stream management,” The VLDB Journal, vol. 12, no. 2, pp. 120–139, 2003.

[11] P. Covington, J. Adams, and E. Sargin, “Deep neural networks for YouTube recommendations,” in *Proceedings of the 10th ACM Conference on Recommender Systems (RecSys)*, 2016, pp. 191–198.

[12] A. Criminisi, J. Shotton, and E. Konukoglu, “Decision forests: A unified framework for classification, regression, density estimation, manifold learning and semi-supervised learning,” *Foundations and Trends in Computer Graphics and Vision*, vol. 7, no. 2–3, pp. 81–227, 2012.