Technical Perspective: Succinct Range Filters

Stratos Idreos
Harvard University

Data structures that filter data for point or range queries are prevalent across all data-driven applications, from analytics to transactions, and modern machine learning applications. The primary objective is simple: find whether one or more data items exist in the database. Yet, this simple task is exceptionally hard to perform efficiently, and surprisingly critical for the overall properties of the data-intensive applications that rely on filtering.

This is a hard problem as there are numerous critical parameters and trade-offs. Many parameters come from the workload, e.g., the exact percentage of point queries versus updates, percentage of empty-result queries, etc. Other parameters come from the underlying hardware; e.g., filters typically reside in memory but, with exponentially increasing data sizes, we need to be mindful of the filter size and the memory hierarchy. Overall, there are complex trade-offs to navigate: memory, read, and write amplification. For example, a data structure cannot be efficient for both point and range queries while also supporting efficient writes. Yet, numerous applications need to expose both read patterns.

A prototypical application of filters is LSM-tree storage engines. An LSM-tree stores data in the order they arrive in immutable files and periodically sort-merges them into larger files. This way, it behaves in between a log and a sorted array, providing a good balance of read and write performance depending on the exact tuning (file size, buffer size, etc.). LSM-tree storage engines are used as the backbone of most distributed key-value stores and applications range from social media, web-applications, e-shopping, IoT, etc. Due to their multi-level architecture enforcing a global temporal order, LSM-tree engines rely heavily on in-memory filters.

In ACM SIGMOD 2018, Succinct Range Filters (SuRF) was introduced as a new succinct filter that can handle point queries, range queries, and approximate counts efficiently [1]. SuRF is based on a trie-like structure termed Fast Succinct Trie. The trie-based design allows building a structure that can support performant range queries and point queries.

The authors of SuRF make the following critical and insightful observation which brings everything together and allows SuRF to balance the various hardware and workload trade-offs. For a given set of queries, the upper levels of the trie incur many more accesses than the lower levels. For this reason, the SuRF design utilizes a dense, performance-optimized encoding scheme for the top of the trie and a sparse, memory-optimized encoding scheme for the bottom. This results in a data structure that is both fast and memory efficient. The upper levels, which are comprised of few nodes but incur many accesses, encode keys under the LOUDS-Dense scheme, sacrificing space efficiency for fast lookups. The lower levels, which contain the majority of nodes but have a sparse access pattern relative to high levels are encoded with LOUDS-Sparse, sacrificing fast lookups for space efficiency.

Compared to state-of-the-art bloom filter based solutions (e.g., prefix bloom filters) SuRF provides a general solution, i.e., it can support any range query as well as efficient point queries. Compared to state-of-the-art tree or trie based solutions SuRF offers similar or better performance at a much smaller memory footprint. The SuRF paper shows end-to-end impact by integrating SuRF in RocksDB, the most mature LSM-tree based storage engine, and demonstrating strong results (e.g., up to 5x) in time-series applications for both point and range queries. SuRF can be applied broadly to any application that needs a succinct filter such as monitoring, privacy/security, graph analytics, etc. Finally, the core spirit of the design of SuRF exemplifies elegant research taste in pursuing hybrid, hardware- and workload-conscious designs.

1. REFERENCES

[1] H. Zhang, H. Lim, V. Leis, D. G. Andersen, M. Kaminsky, K. Keeton, and A. Pavlo. SuRF: Practical Range Query Filtering with Fast Succinct Tries. In ACM SIGMOD 2018.