CAS CS 561: Data Systems Architectures
Data-intensive Systems and Computing Lab
Department of Computer Science
College of Arts and Sciences, Boston University
http://bu-disc.github.io/CS561/

CS561 Spring 2024 - Research Project

Title: *Project Qu-ART Tree – QuIT for ART?*

Background: The Adaptive Radix Tree (ART)[1] is a trie-data structure offering efficient main-memory indexing. ART builds upon the radix tree data structure that offers operations at an $O(k)$ complexity where k denotes the length of the key and is independent of n, the number of entries. However, ART offers differently sized nodes, tuned for exploiting modern hardware characteristics. Adapting each index/internal node locally allows ART to optimize for space and efficiency at the same time.

Meanwhile, recent work on Sortedness-Aware indexing data structures[2,3] have focused on harnessing inherent data characteristics, i.e., sortedness to improve indexing performance. QuIT or *Quick Insertion Tree* proposes two interesting strategies to adapt to sortedness – *lil* and *pole*, while also discussing pitfalls of a naïve strategy (*tail-leaf insertions*) that has been widely adapted in popular data systems.

Objective: This project aims to explore the feasibility of applying QuIT-like principles to ART. Particularly, since ART is a trie-based data structure, moving fast-path pointers to parent-nodes (capitalizing prefixes to be more nearly-sorted) can yield more benefits when compared to moving the pointers across the sibling nodes.

Technical: This project requires C++ programming skills (particularly working with pointers). We can start off with an existing implementation of ART to integrate and test our changes:

1. Implement tail-leaf/tail-inner node optimization + benchmark against standard ART.
2. Implement *lil* in ART, which essentially will be a pointer to the last-inserted internal node of the tree + another variant that will move the pointer upwards to its parent rather than between siblings.
3. Benchmark both versions of *lil* against other baselines.
4. Explore *pole* for ART – how would its estimator work in a trie-data structure, any possibility of optimizations, etc.
5. Implement *pole* in ART + benchmark

Responsible Mentor: *Aneesh Raman*

References:

[1] *Viktor Leis, Alfons Kemper, and Thomas Neumann. 2013. The adaptive radix tree: ARTful indexing for main-memory databases. In Proceedings of the 2013 IEEE International Conference on Data Engineering (ICDE 2013) (ICDE '13). IEEE Computer Society, USA, 38–49. https://doi.org/10.1109/ICDE.2013.6544812*

[2] *A. Raman, S. Sarkar, M. Olma and M. Athanassoulis, "Indexing for Near-Sorted Data," 2023 IEEE 39th International Conference on Data Engineering (ICDE), Anaheim, CA, USA, 2023, pp. 1475-1488, doi: 10.1109/ICDE55515.2023.00117.*

[3] *A. Raman, K. Karatsenidis, S. Xie, M. Olma, S. Sarkar, M. Athanassoulis, "QuIT your B⁺-tree for the Quick Insertion Tree", 2025 28th International Conference on Extending Database Technology (EDBT) (EDBT '2025), Barcelona, Spain.*