HB+Trie

Thushjandan & François-Xavir

June 02, 2022

Data Management Data Structures

Agenda

1. Motivations
2. Overview
3. Implementation
4. Performance
5. Possible improvements
6. Discussion

Motivations

Motivations

Variable-length sized keys

Disadvantages with B+ tree or LSM-tree:

- Fanout degree decreases if key length increases
- Tree Height grows to maintain the same capacity
- Benefit of prefix B+ tree becomes limited for randomly distributed keys
- B+ tree nodes are randomly scattered on disk when it ages

Overview

Overview

HB+ trie stands for *Hierarchical B+ tree based trie*

Characteristics:

- Key space is divided into buckets. Every bucket has its own HB+ trie
- High disk throughput due to append-only disk layout
- Disk updates are delayed with a **Write buffer index**

Overview

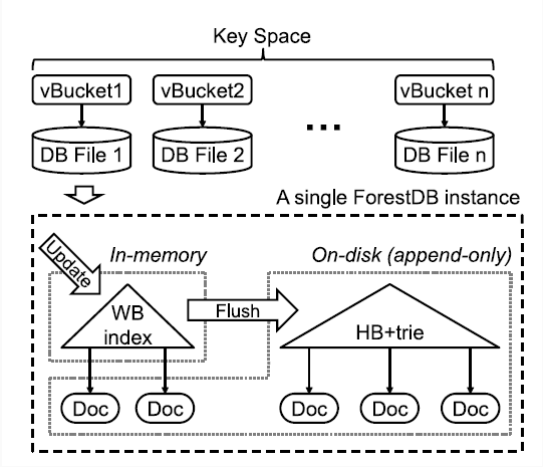


Figure 1: Architecture

Overview

HB+ trie stands for *Hierarchical B+ tree based trie*

Characteristics:

- Key space is divided into buckets. Every bucket has its own HB+ trie
- High disk throughput due to append-only disk layout
- Disk updates are delayed with a **Write buffer index**
- Fixed size chunking of the key
- Every unique chunk has a dedicated B+ tree

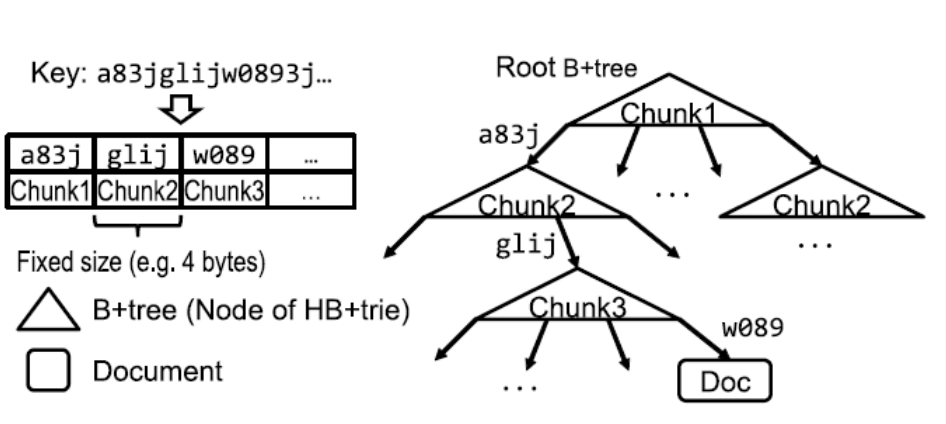


Figure 2: Chunking

Overview

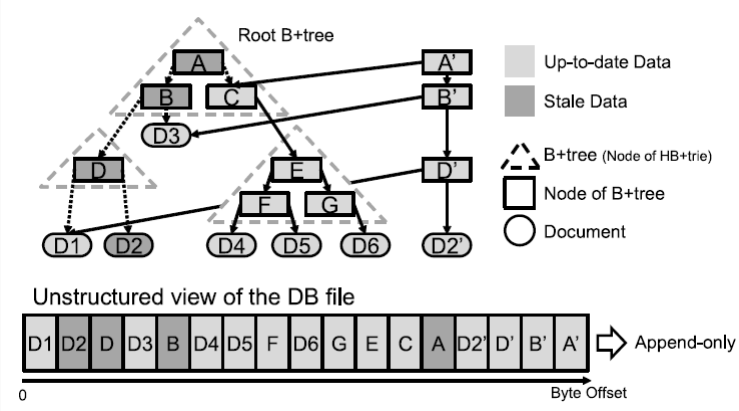


Figure 3: Disk layout

Implementation

Implementation

- Using 16 byte chunks for keys
- Each page frame holds a complete B+ subtree.
- Storing pageId in the leaf to reference a B+ subtree

Performance

Performance

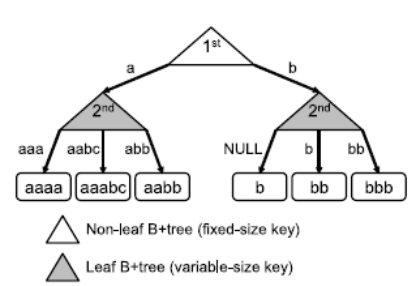
Possible improvements

Possible improvements

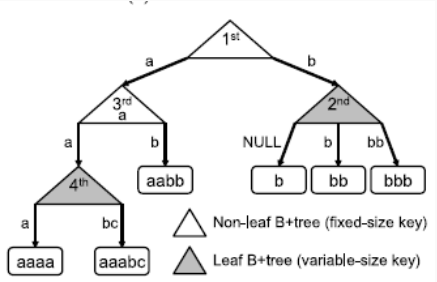
HB+ trie is not a balanced structure

- Leads to key skew under specific key pattern

To address this issue, Leaf B+ tree extension is proposed



(a) Without extension



(b) With extension

Discussion