TreeLine: An Update-In-Place Key-Value Store for Modern Storage

저자 및 학회

2024. 01. 16

Presentation by Yeojin Oh, Jisoo Lee, Zhu Yongjie

yeojinoh@dankook.ac.kr, lkhejj1@gmail.com,

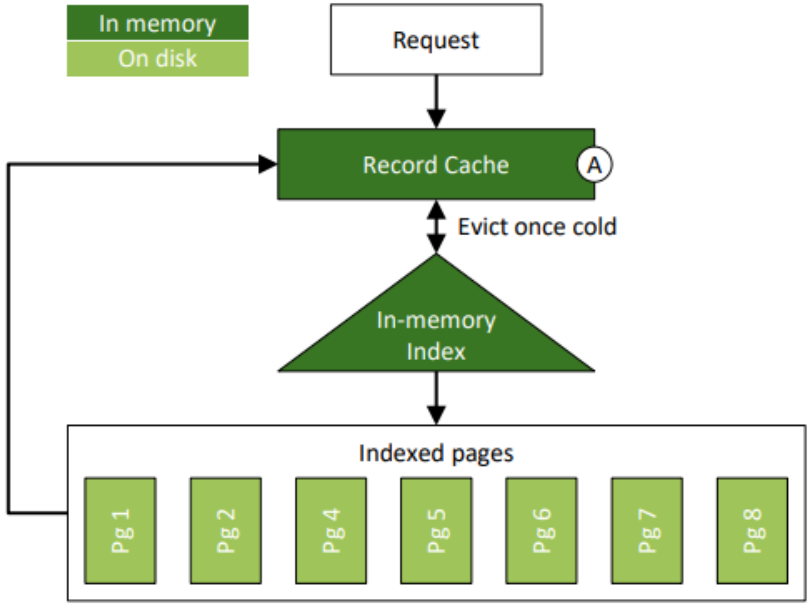
Contents

1. Structure of TreeLine

- **What is TreeLine?**
  - An update-in-place key-value store.
  - Well-balanced between read and write performance.
  - Highly efficient in large-scale database systems.

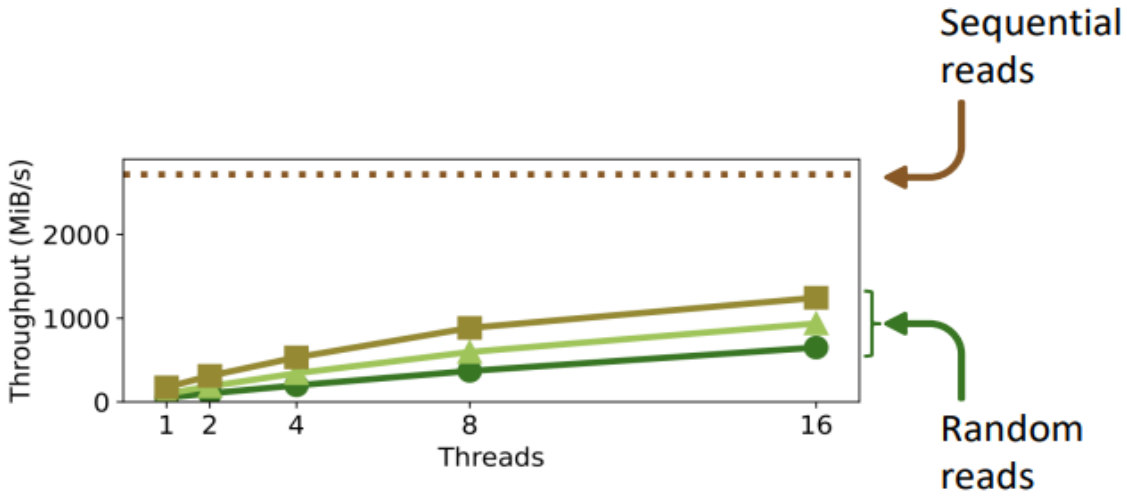
1. Structure of TreeLine

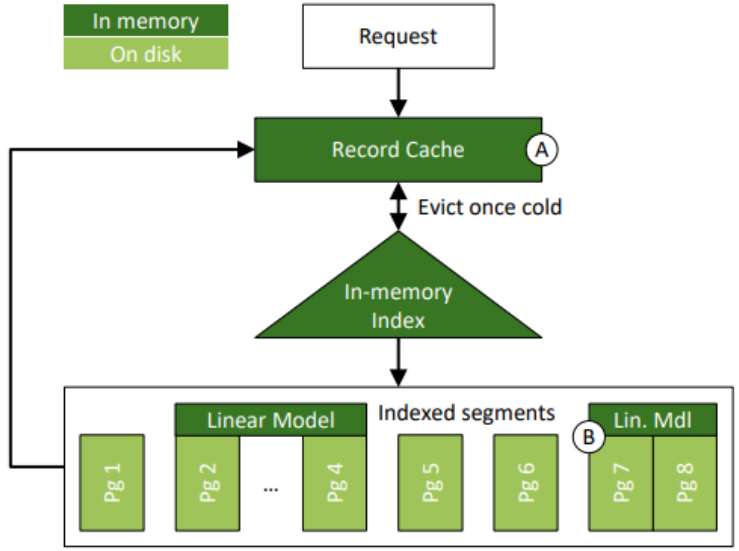
- **Record Caching**
  - Optimizes write operations using in-memory caching
  - Enhances read performance by maximizing the retention time of hot records in memory



1. Structure of TreeLine

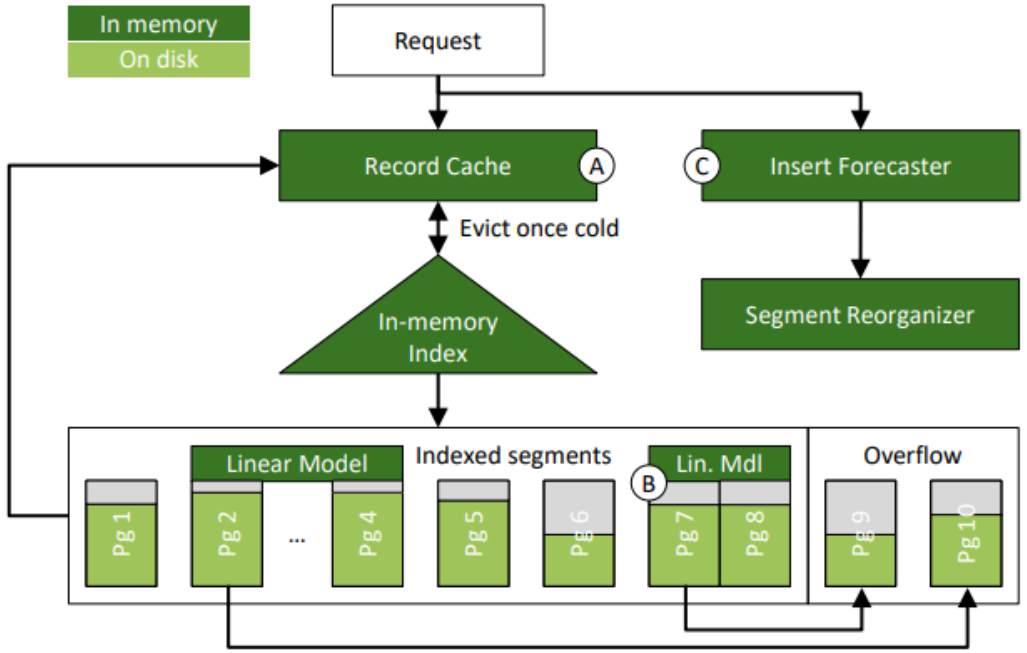
- **Page Grouping**
  - Groups pages that store adjacent keys
  - Stores grouped pages consecutively to improve access efficiency





1. Structure of TreeLine

- **Insert Forecasting**
  - Analyzes insertion patterns to predict future insert loads for intelligent space management



2. Building a TreeLine Environment

- TreeLine is an open-source project on GitHub.
  - git clone https://github.com/mitdbg/treeline.git
  - Dependencies
    - libtbb-dev
    - autoconf
    - libjemalloc-dev
    - CMake3.17+

apt install libtbb-dev autoconf libjemalloc-dev

Create a virtual Python environment using anaconda and "pip install cmake==3.17"

2. Building a TreeLine Environment

- TreeLine is an open-source project on GitHub.
  - Compile
    - mkdir build && build
    - cmake -DCMAKE_BUILD_TYPE=Release .. && make –j
    - cmake -DCMAKE_BUILD_TYPE=Release -DTL_BUILD_TESTS=ON .. && make –j
    - cmake -DCMAKE_BUILD_TYPE=Release -DTL_BUILD_BENCHMARKS=ON .. && make –j

2. Building a TreeLine Environment

- TreeLine BenchMark
  - YCSB
    - Go to the " /path/to/db/treeline/build/bench " directory and "./run_custom"
    - The workload configuration file is located in " /tree/bench/workload_configs/ "
    - Run " ./run_custom –workload_config= /path/to/db/tree/bench/workload_configs/phased_64B_A_B_A_B_A.yml "
  - Other Bench (adjust options)
    - Temporarily not found

3. Future Experimental Plans

- How does TreeLine compare against RocksDB and LeanStore on throughtput and the amount of physical I/O performed?
- How do the record cache and page grouping contribute to TreeLine's overall performance?
- How does the choice of the page grouping parameters affect the grouping "effectiveness"?
- How effective is insert forecasting?

3. Future Experimental Plans

- YCSB workload

Workload	Description
A	50% Read, 50% Update
B	95% Read, 5% Update
C	100% Read
D	95% Read Latest, 5% Insert
E	95% Range Scan (average length 50, maximum length 100), 5% Insert
F	50% Read-Modify-Write, 50% Read

Thank you