University of Zurich[UZH]

Department of Informatics

University of Zürich
Department of Informatics
Binzmühlestr. 14
CH-8050 Zürich
Phone. +41 44 635 43 11
Fax +41 44 635 68 09
www.ifi.uzh.ch/dbtg

UZH, Dept. of Informatics, Binzmühlestr. 14, CH-8050 Zürich

Prof. Dr. Michael Böhlen
Professor
Phone +41 44 635 43 33
Fax +41 44 635 68 09
boehlen@ifi.uzh.ch

Zürich, 24. August 2017

BSc Vertiefungsarbeit
Topic: Detecting Volatile Index Nodes in a Hierarchical Database System

Apache Jackrabbit Oak[1] is a hierarchical database system. It organizes all data in a single tree and since the hierarchical model naturally captures the structure of webpages, Oak is the basis for several CMSs (e.g., Adobe Experience Manager, Magnolia, etc.). A typical CMS workload consists of modifying and publishing webpages. This workload is write-heavy and skewed, since modifications are common and some webpages are updated more frequently than others. Publishable webpages are indexed and the described workload causes the same index nodes to be repeatedly inserted/deleted. We call these index nodes volatile.

Volatile index nodes raise two main issues. First, since index modifications propagate up and down the tree, inserting/deleting a volatile index node causes a sequence of index nodes to be inserted/deleted. The index update performance consequently deteriorates, because volatile index nodes and their ancestors are repeatedly inserted/deleted. Second, transactions are likely to conflict with one another when they concurrently insert/delete volatile index nodes that share a common ancestor on the path to the root node. To resolve the conflict, one of the two involved transactions needs to abort. Aborting and restarting transactions is expensive and limits the transactional throughput.

In this project the student should understand the problems caused by volatile index nodes and study the approach presented in [1] to deal with them. The main goal of this project is twofold. First, the student should familiarize with the architecture and code of the database system Apache Jackrabbit Oak. Second the student should implement the techniques presented in [1] in Oak's kernel.

[1] `https://jackrabbit.apache.org/oak/`

Tasks

1. Study and understand [1].
2. Implement the following components in Apache Jackrabbit Oak as described in [1]. The implementation should be based on the document-based storage back-end (MongoDB).
   (a) Computing the volatility count of a node and deciding if it is volatile
   (b) Workload-aware pruning of index nodes
   (c) Document splitting
3. Summarize your work in a short report (approximately 10 pages).

Optional Task

1. Evaluate your implementation based on a synthetic dataset. Choose the dataset to evaluate the technique in worst-case and best-case scenarios.

References

[1] K. Wellenzohn, M. Böhlen, S. Helmer, M. Reutegger, and S. Sakr. A Workload-Aware Index for Tree-Structured Data. To be published.

Supervisor: Kevin Wellenzohn (wellenzohn@ifi.uzh.ch)

Start date: 5 September 2017

University of Zurich
Department of Informatics

Prof. Dr. Michael Böhlen
Professor