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, 9. Oktober 2017

BSc Thesis
Topic: An Adaptive Index for Hierarchical Distributed Database Systems

Apache Jackrabbit Oak[1] is a hierarchical distributed 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 to webpages 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: (a) repeatedly inserting/deleting index nodes and their ancestors is expensive, and (b) conflicts between concurrent transactions get more likely as the level of contention increases.

The Workload-Aware Property Index (WAPI) [1] manages the volatility in the index. If an index node n is volatile based on the recent workload, WAPI does not prune n anymore. When the workload characteristics change, new index nodes can become volatile while others cease to be volatile and become *unproductive*. Unproductive index nodes slow down queries as traversing an unproductive node is useless, because neither the node itself nor any of its descendants contain an indexed property and thus cannot yield a query match. Additionally, unproductive nodes occupy storage space that could otherwise be reclaimed. If the workload changes frequently, unproductive nodes quickly accumulate in the index and the query performance deteriorates over time. Therefore, unproductive nodes must be cleaned up to keep query performance stable over time and reclaim disk space as the workload changes.

We propose two approaches to deal with unproductive index nodes. The first approach (used as a simple baseline comparison) is a periodic Garbage Collection (GC). The algorithm traver-

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

ses the full index subtree and prunes all unproductive nodes at once. The second approach, called Query-Time Pruning (QTP), is an incremental approach to cleaning up unproductive nodes in the index. The idea is to turn queries into updates. Since Oak already traverses unproductive nodes as part of query processing, these nodes could be pruned at the same time. In comparison to GC, with QTP only one query has to traverse an unproductive node, while subsequent queries can skip this overhead and thus perform better.

The goal of this BSc thesis is to study, implement, and empirically compare GC and QTP. The student should implement both approaches in the kernel of Apache Jackrabbit Oak. The experimental evaluation, detailed below, shall compare GC and QTP under changing workloads.

Tasks

1. Study and understand [1] with a focus on what unproductive nodes are and how [1] proposes to deal with them.
2. Implement Garbage Collection (GC) in Apache Jackrabbit Oak. The periodicity of GC should be customizable.
3. Implement Query-Time Pruning (QTP) in Apache Jackrabbit Oak.
4. Evaluate and compare GC and QTP empirically under changing workloads. Choose appropriate metrics to study in your experimental evaluation and explain why they are relevant. The experiments should be conducted with different workloads and datasets that stress-test the algorithms. You can assume a serial execution of transactions, i.e., transactions cannot be executed concurrently. Your empirical evaluation should address the following questions:
   (a) What is the performance penalty incurred by unproductive nodes on query response times for different values of WAPI's two parameters: threshold τ and sliding window length L?
   (b) What is the cost of GC? How does GC's periodicity affect query response times?
   (c) What is the overhead incurred by QTP on query execution? In particular, how expensive is it to check for a node's productiveness and how expensive is it to prune unproductive nodes?
   (d) How does the skewness and query-update ratio of the workload affect query response times for GC and QTP?
5. Write the thesis (approximately 50 pages).
6. Present the thesis in a DBTG meeting (25 minutes presentation).

Optional Tasks

1. Simulate concurrent transactions in your experimental evaluation. How does concurrency (negatively) affect GC and QTP? In particular, how do the query response times and the number of transaction aborts change for increased levels of concurrency?

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: 9 October 2017

End date: 1 February 2018

University of Zurich
Department of Informatics

[illegible]

Prof. Dr. Michael Böhlen
Professor