An Adaptive Index for Hierarchical Database Systems

Rafael Kallis

BSc Thesis

February 3, 2018

University of Zurich[UZH]

Department of Informatics

The Workload-Aware Property Index (WAPI):

- Detects frequently updated nodes
- Stops pruning such volatile nodes
- Significantly improves update throughput

Unproductive Nodes are an unwanted byproduct:

- When the workload changes, volatile nodes cease to be volatile
- Then they waste space and slow down queries
- They do not contribute to a query match and contain no data

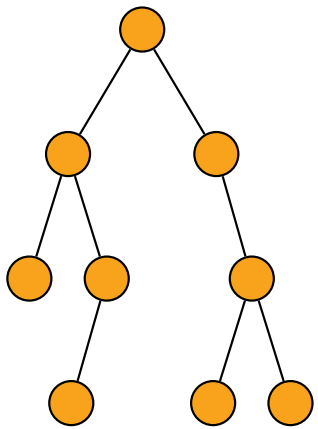
In this thesis we:

- Design and implement two solutions in order to mitigate unproductive nodes
- Analyze the factors impacting the production of unproductive nodes
- Empirically evaluate and compare our two solutions

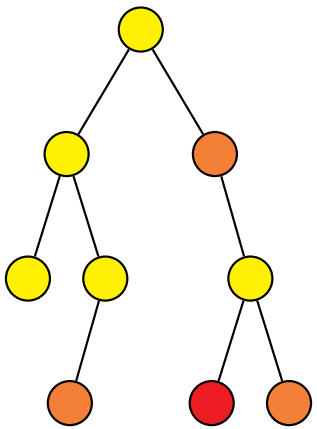
A Content Management System's (CMS) workload is:

- skewed
- update-heavy
- changing

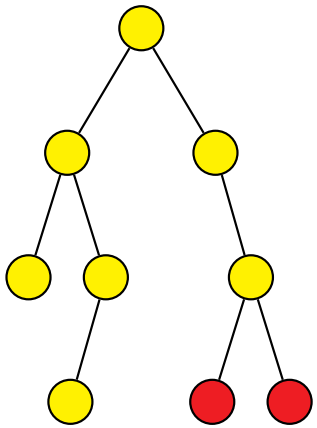
Skewed Workload: small subset of nodes gets frequently updated



No skew

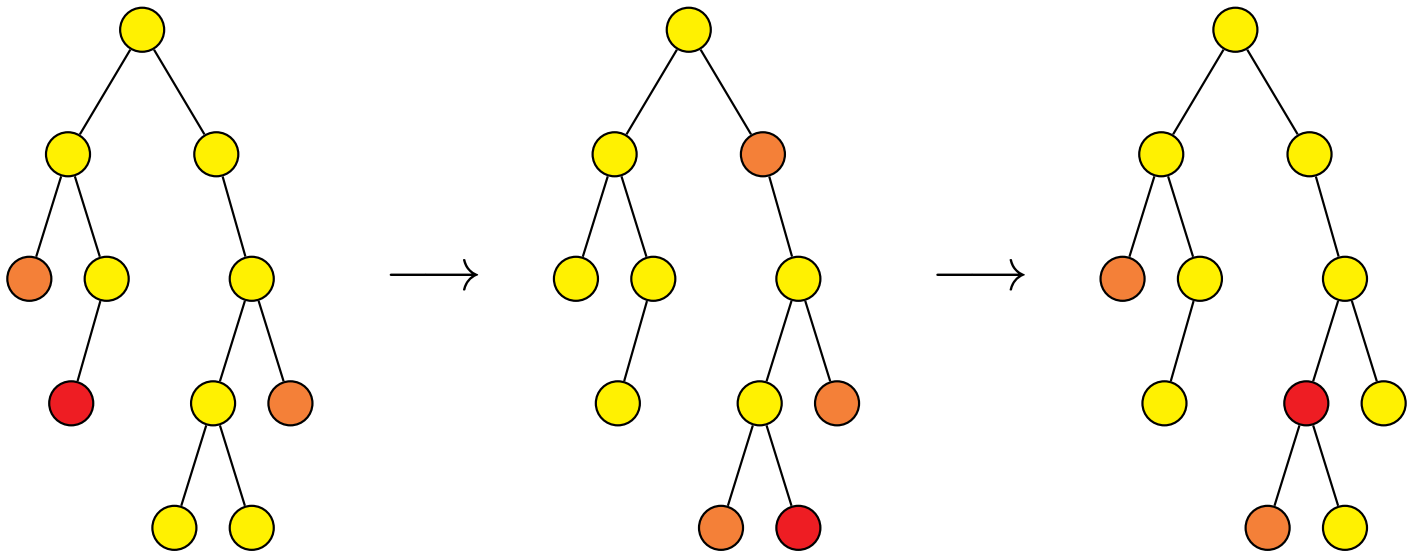


Normal skew



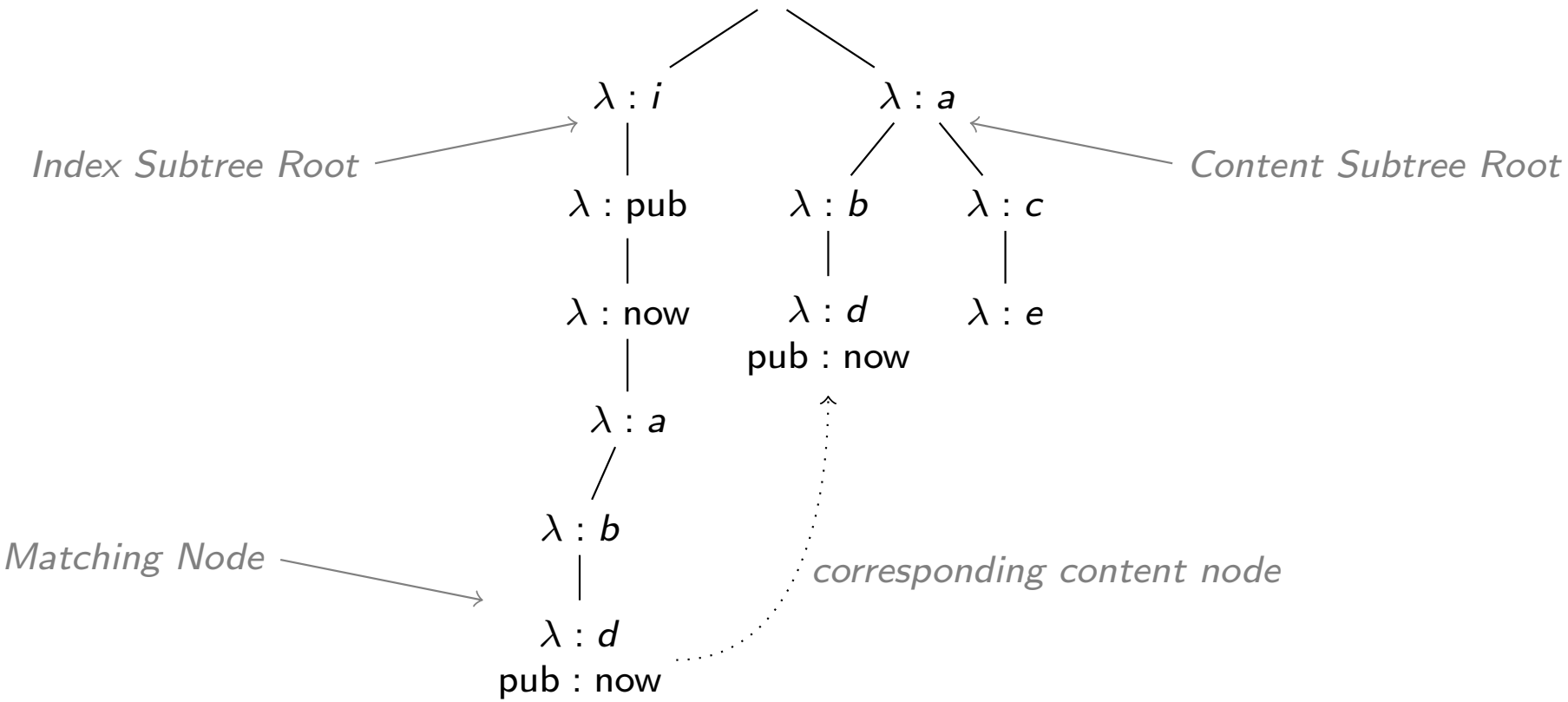
High skew

Changing workload: as time passes, hotspots change



CMSs usually use a job-queuing system that has the notes characteristics

Hierarchical Database with WAPI



Oak mostly executes content-and-structure (CAS) queries [1]. We denote node n's property k as $n[k]$ and node n's descendants as $desc(n)$.

Definition (CAS Query)

Given node m, property k and value v, a CAS query $Q(k, v, m)$ returns all descendants of m which have k set to v, i.e.,

$$Q(k, v, m) = \{n | n \in desc(m) \wedge n[k] = v\}$$

Volatility is the measure which is used by the WAPI in order to distinguish whether to remove a node or not from the index. Wellenzohn et al. [2] propose to look at the recent transactional workload to check whether a node n is volatile.

Definition (Volatility Count)

The volatility count $vol(n)$ of index node n on database instance O_i, is the number of times node n was added or removed from snapshots contained in a Sliding Window of Length L over history H_i, i.e.,

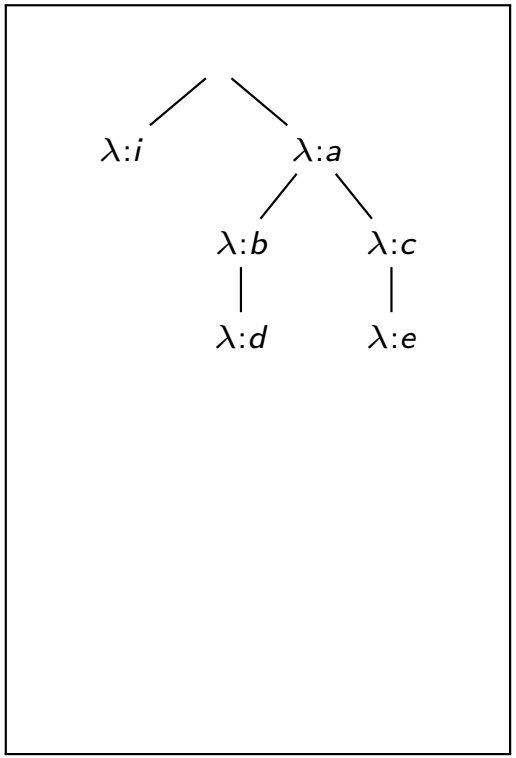
$$vol(n) = |\{G^b | G^b \in H_i \wedge t(G^b) \in [t_n - L + 1, t_n] \wedge \exists G^a [G^a = pre(G^b) \wedge ([n^a \notin N(G^a) \wedge n^b \in N(G^b)] \vee [n^a \in N(G^a) \wedge n^b \notin N(G^b)])]\}|$$

Definition (Volatile Node)

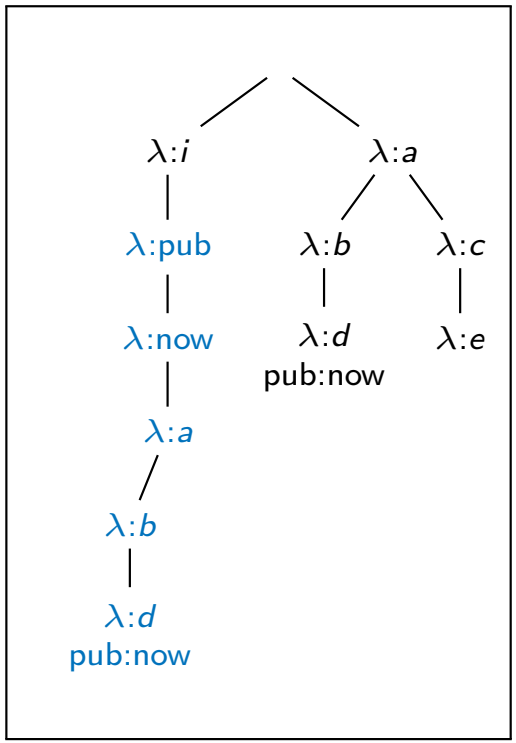
Index node n is volatile iff n's volatility count is greater or equal than the volatility threshold τ, i.e.,

$$volatile(n) \iff vol(n) \geq \tau$$

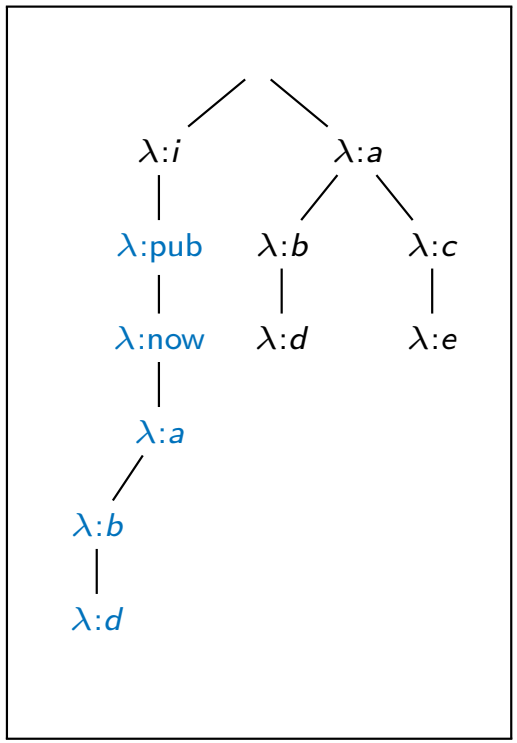
$$G^0 \xrightarrow{T_1} G^1 \xrightarrow{T_2} G^2$$



Snapshot G^0
$t(G^0) = t$



Snapshot G^1
$t(G^1) = t + 1$



Snapshot G^2
$t(G^2) = t + 2$

frame title

Block title

Block contents

- item1
- item2

References

Mathis, C., Härder, T., Schmidt, K., and Bächle, S.
XML indexing and storage: fulfilling the wish list.
Computer Science - R&D 30, 1 (2015), 51–68.

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