UNIVERSITÄT PADERBORN
Die Universität der Informationsgesellschaft
Faculty for Computer Science, Electrical Engineering and Mathematics
Department of Computer Science
Research Group Data Science

Master's Thesis

Submitted to the Data Science Research Group
in Partial Fullfilment of the Requirements for the Degree of

Master of Science

Space Reduction of Tentris Hypertrie with Path Compression

by
Burhan Otour

Thesis Supervisors:
Prof. Dr. Axel-Cyrille Ngonga Ngomo
Prof. Dr. Stefan Böttcher

Paderborn, August 10, 2020

Erklärung

Ich versichere, dass ich die Arbeit ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Quellen angefertigt habe und dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen worden ist. Alle Ausführungen, die wörtlich oder sinngemäß übernommen worden sind, sind als solche gekennzeichnet.

Ort, Datum	Unterschrift

Abstract. In my thesis, I investigate and implement a space reduction approach for the in-memory indexing data structure Hypertrie.

Contents

1 Introduction

2 Preliminaries and Foundations

2.1 C++ and Meta-programming

2.2 Pointer Tagging

2.2.1 Data Structure Alignment

2.2.2 Tagged Pointers

2.3 Trie-based Data Structures

2.4 Tentris

2.4.1 Mapping RDF to Tensor

2.4.2 Mapping Tensor to Hypertrie

3 Related Work

asdasdasd

4 Space Reduction Approach

This part of the thesis discusses the approach to substantially mitigates the space inefficiency characteristic of Hypertrie. The technique relies mainly on compressing a Hypertire path with specific characteristics. Worth mentioning that the approach does not neglect the other attempts already realized to minimize Hypertrie memory footprint. In contrast, it can be considered an added feature that further contributes to the space reduction of the overall Hypertrie data structure.

oter structure
sit"

In this chapter, I deliver a motivation to the approach. Afterward, I discuss the new Hypertrie internal nodes' design needed to realize the path compression feature. Finally, algorithms defining the behaviors of the newly designed Hypertrie are also presented.

4.1 Motivation

Despite its operational efficiency, Hypertrie performance comes not without a trade-off. Since Hypertrie is a special kind of a Trie data structure, it inherits some of the fundamental problems of Tries. One of these problems is the excessive space utilization in a worst-case scenario.

The current design and implementation of Hypertrie, however, mitigates the space inefficiency characteristic in two ways. First, for each tensor dimension mapping in each node, the Hypertrie uses custom map data structures instead of arrays or linked lists to store the keys. By using a map, Hypertrie's nodes only stores keys that form prefixes to already existed paths. In contrast, arrays utilization in normal Tries considers the whole alphabet set in each node with many array entries store pointers that refer to null.

The adoption of maps in Hypertrie also delivers extra performance as looking up keys in a carefully designed map is nearly constant compared to linked list search where it has a linear complexity $O(n)$. The other solution realized by Hypertrie to reduce the overall space requirement is to store equal nodes (Subhypertrie) only once. In this way, Hypertrie achieves a moderate level of compression in practice.

Despite the previously mentioned attempts to minimize the size of Hypertrie, the excessive memory requirement is still a bottleneck. The case can be witnessed when the set of RDF triples needed to be indexed by Hypertrie increases in size with less overlapping between its elements. As a result, many intermediate nodes store map with a single entry for a particular dimension where the entry hosts a space for key and a pointer. This becomes a space redundancy issue when the leaf node referenced by the pointer has one key only.

The purpose of the following approach is to try to reach a more space-efficient Hypertrie. Continue here

4.2 Compressed Hypertrie Node Representations

In order to achieve path compression in Hypertrie, fundamental design changes need to take place. By that, we can enable the node to store the entire key suffix. Concretely, each group of define key suffi Hypertrie nodes in certain tree depth will have their own internal node representation.

From programming point of view, the redesign of Hypertrie nodes' structures is low level. Thanks to C++17 template meta-programming feature, we could separate the compressed nodes realization from the Hypertrie data structure interface. By that, we can still insure a smooth integrity of Hypertrie with other components in Tentris system.

Depth 3 Node

Depth 2 Node

Depth 1 Node

4.2.1 Node Expansion

4.3 Algorithms

]

5 Evaluation and Benchmarking

asdasdasd

6 Conclusion

asdasdasd

Bibliography

[OPHS16] Tobias Oetiker, Hubert Partl, Irene Hyna, and Elisabeth Schlegl. The Not So Short Introduction To $\LaTeX\,2_\varepsilon$, 2016. Checked 2017-12-18.

[The17] The CTAN Team. CTAN Comprehensive $\TeX$ Archive Network, 2017. Checked 2017-12-18.