SIMD Acceleration for Index Structures

Marten Wallewein-Eising
Otto-von-Guericke University
Magdeburg, Germany
marten.wallewein-eising@st.ovgu.de

***Abstract*—**

- **summary:**
  Give short an overview of SIMD and modern index structures
  Explain what are the problems of the "old" index structures made for disk-based database systems
  Explain which approaches were made to adapt index structures to modern systems and what they have in common and what are differences
- **Why is this work important:**
  Give a state of current development of the index structures
  Collect common approaches to adapt other index structures TODO: ReThink
- **K-ary search trees, FAST, VAST and ART compared**
- **Contribution: What are important approaches used by different implementations to adapt index structures to modern systems**

***Index Terms*—SIMD, index**

I. Introduction

- General Problem: Index structures are not applicable to modern systems
  IO-Bottleneck moves from disk-ram to ram-cache. CPU cycles of pure calculation got more important, cache line has to be used in an optimized way
  Index structures grow very high because of the massive amount of data collected in modern databases
  Branch-mispredictions cost many CPU cycles and should therefore be avoided
- objectives/contributions: Comparision of current work and highlighting of important approaches for index structures
- main result: Adapting of tree nodes to cache line and SIMD blocks and use small keys to compare as much keys as possible with one SIMD instruction

II. Preliminaries

A. *Simple instruction multiple data*

Short overview of the following:

- How SIMD works
- SIMD vs vertical vector processing

B. *Considered index structures*

Short overview of the following: TODO: Maybe the problems of these old index structures?

- Binary tree???
- B^+ tree
- Radix tree

III. Adapted Tree Structures

TODO: Compare all 4 or merge FAST and VAST together/ extend FAST with VAST??

A. *K-ary search tree*

B. *FAST*

C. *ART*

D. *VAST*

IV. Evaluation

In common:

- SIMD instructions used to compare the search key with multiple keys of the index
- Segmenting tree to blocks for a better usage of cache lines, save the data of the nodes in an adapted way
- The keys should be as short as possible to compare more keys in one step and to decrease the passed data to the cache line
- Each approach improves the tree traversal

Differences:

- Node compression in VAST, Path compression in ART and K-ary seg trie
- FAST and K-ary trees readonly to improve traversal, ART and FAST adapt insert too
- FAST uses and K-ary trees will use GPU calculation instead of CPU

Why performance can not be compared in a useful way...

V. Related Work

TODO:

- ART and VAST compared to FAST??
- Ideas and implementations of the adapted trees already in III...

VI. Future work

???

References

[1] Mohammad Suaib, Abel Palaty and Kumar Sambhav Pandey, "Architecture of SIMD Type Vector Processor" in International Journal of Computer Applications (0975 - 8887) Volume 20 No.4, April 2011.

[2] Jingren Zhou and Kenneth A. Ross "Implementing Database Operations Using SIMD Instructions" in ACM S1GMOD '2002 June 4-6, Madison, Wisconsin, USA

[3] Steffen Zeuch, Frank Huber and Johann-Christoph Freytag "Adapting Tree Structures for Processing with SIMD Instructions" in Proc. 17th International Conference on Extending Database Technology (EDBT), March 24-28, 2014, Athens, Greece

[4] Viktor Leis, Alfons Kemper and Thomas Neumann "The Adaptive Radix Tree: ARTful Indexing for Main-Memory Databases" in ???
[5] Takeshi Yamamuro, Makoto Onizuka,Toshio Hitaka, and Masashi Yamamuro "VAST-Tree: A Vector-Advanced and Compressed Structure for Massive Data Tree Traversal" in EDBT 2012, March 26-30, 2012, Berlin, Germany.
[6] Changkyu Kim, Jatin Chhugani, Nadathur Satish, Eric Sedlar, Anthony D. Nguyen, Tim Kaldewey, Victor W. Lee, Scott A. Brandt and Pradeep Dubey "FAST: Fast Architecture Sensitive Tree Search on Modern CPUs and GPUs" in SIGMOD10, June 6-11, 2010, Indianapolis, Indiana, USA. TODO: Author count more than 6