OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG

INF FAKULTÄT FÜR INFORMATIK

SIMD Acceleration for Index Structures: A Survey

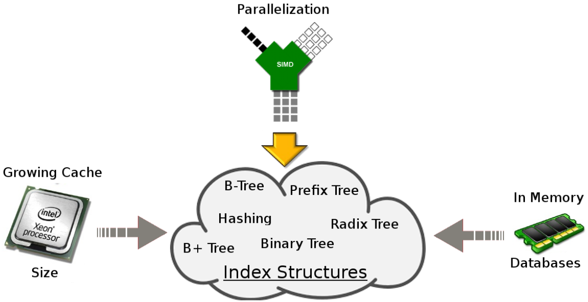
Marten Wallewein-Eising

Otto von Guericke University, Magdeburg

September 11, 2018

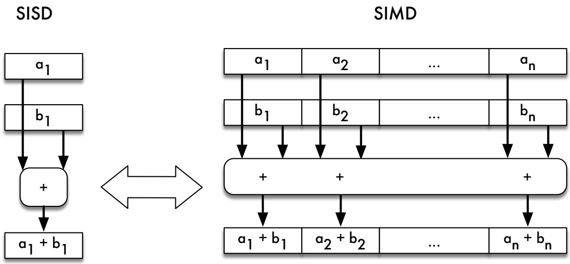
Agenda

Motivation



- What are state-of-the-art index structures?
- Which optimizations do they have in common?

Single Instruction Multiple Data

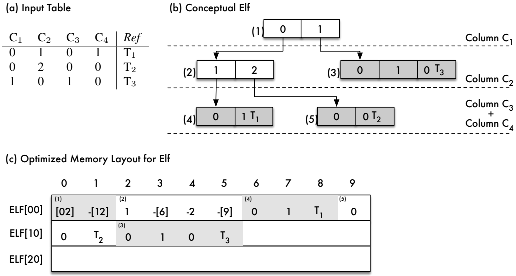


- **__m128i _mm_add_epi32 (__m128i a, __m128i b)** Adds 4 signed 32-bit integers in a to 4 signed 32-bit integers in b.

Surveyed Index Structures

- Elf
- Seg-Tree/Trie
- FAST: Fast Architecture Sensitive Tree
- VAST: Vector-Advanced and Compressed Structure Tree
- ART: Adaptive Radix Tree

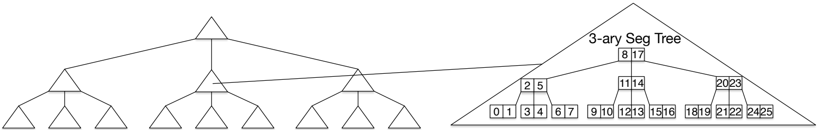
Elf



- Multi-dimensional index structure for column-wise storage
- Prefix-redundancy elimination on distinct column values
- Linearisation for optimized memory layout

[Broneske et al. ICDE'17]

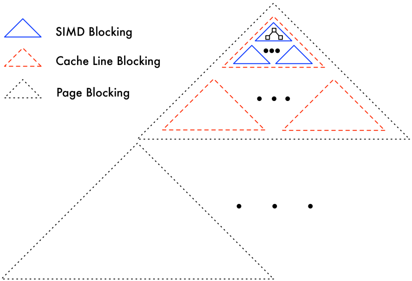
Seg-Tree/Trie



- Each node is a k-ary search tree
- Each node is linearised to use k-ary search
- $k = \frac{|SIMD|}{|Key|}$ partitions, $k - 1$ separator keys are compared in parallel

[Zeuch et al. EDBT'14]

Fast Architecture Sensitive Tree



- Based on binary tree
- Hierarchical blocking: SIMD, cache line and page blocks
- Efficient register, cache line and page usage

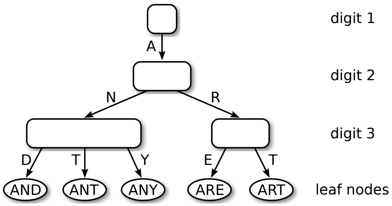
[Kim et al. SIGMOD'10]

Vector-Advanced and Compressed Structure Tree

- Extension of FAST
- Decrease branch misses avoiding conditional branches
- Uses key compression
  - Lossy compression for inner nodes
  - Lossfree compression leaf nodes
- Decompression and error correction of lossy compression has less impact compared to the performance increase with SIMD

[Yamamuro et al. EDBT'12]

Adaptive Radix Tree



- Uses different node types with different number of keys and children
- Due to overfill or underfill of nodes, the node type is changed
- Reduced space consumption due to lazy expansion and path compression

[Leis et al. ICDE'13]

Evaluation

Considered performance criteria for comparing the surveyed index structures:

- Horizontal Vectorization
- Minimized key size
- Adapted node sizes / types
- Decreased branch misses
- Exploit cache lines using blocking and alignment
- Usage of Compression
- Adapt search algorithm for linearized nodes

Evaluation

Implementation of the considered performance criteria and their impact:

Criterion	**Seg-Tree/ Trie**	**FAST**	**ART**	**VAST**	**Elf**	**Impact**
Horizontal vectorization	x	x	x	x	-	high
Minimized key size	o	-	x	x	-	high
Adapted node sizes / types	-	x	-	x	-	low
Decreased branch misses	-	x	-	x	-	medium
Exploit cache lines using blocking and alignment	-	x	-	x	x	medium
Usage of Compression	o	-	x	x	x	medium
Adapt search algorithm for linearized nodes	x	-	-	-	x	low

Legend: x = implements the issue; o = partially implements the issue;
- = does not implement the issue

Conclusion

How to adapt index structures to modern database systems:

- Compare as many keys as possible in parallel with SIMD
  - Direct performance increase up to a multiple
- Efficient usage of cache line
- Decrease branch misses
- Use compression or/and adapted search algorithms

Conclusion

How to adapt index structures to modern database systems:

- Compare as many keys as possible in parallel with SIMD
  - Direct performance increase up to a multiple
- Efficient usage of cache line
- Decrease branch misses
- Use compression or/and adapted search algorithms

Thank you for your attention!

Sources

- `http://infolab.stanford.edu/~nsample/cs245/handouts/hw2sol/sol2.html`
- `https://www.clker.com/clipart-bosque.html`
- Datenbanken Implementierungstechniken, Ausgabe 3, Saake und Sattler

Sources

- T. Yamamuro, M. Onizuka, T. Hitaka, and M. Yamamuro "VAST-Tree: A Vector-Advanced and Compressed Structure for Massive Data Tree Traversal" in EDBT, pp. 396-407, 2012.
- S. Zeuch, F. Huber and J.-C. Freytag "Adapting Tree Structures for Processing with SIMD Instructions" in EDBT, 2014.
- C. Kim, J. Chhugani, N. Satish, E. Sedlar, A. D. Nguyen, T. Kaldewey, V. W. Lee, S. A. Brandt and P. Dubey "FAST: Fast Architecture Sensitive Tree Search on Modern CPUs and GPUs" in SIGMOD, pp. 339-350, 2010.
- V. Leis, A. Kemper and T. Neumann "The Adaptive Radix Tree: ARTful Indexing for Main-Memory Databases" in ICDE, pages 38-49, 2013.