INF FAKULTÄT FÜR INFORMATIK

SIMD Acceleration for Index Structures

Marten Wallewein-Eising

Otto von Guericke Univerity, Magdeburg

September 5, 2018

Agenda

Motivation

SIMD Style Processing

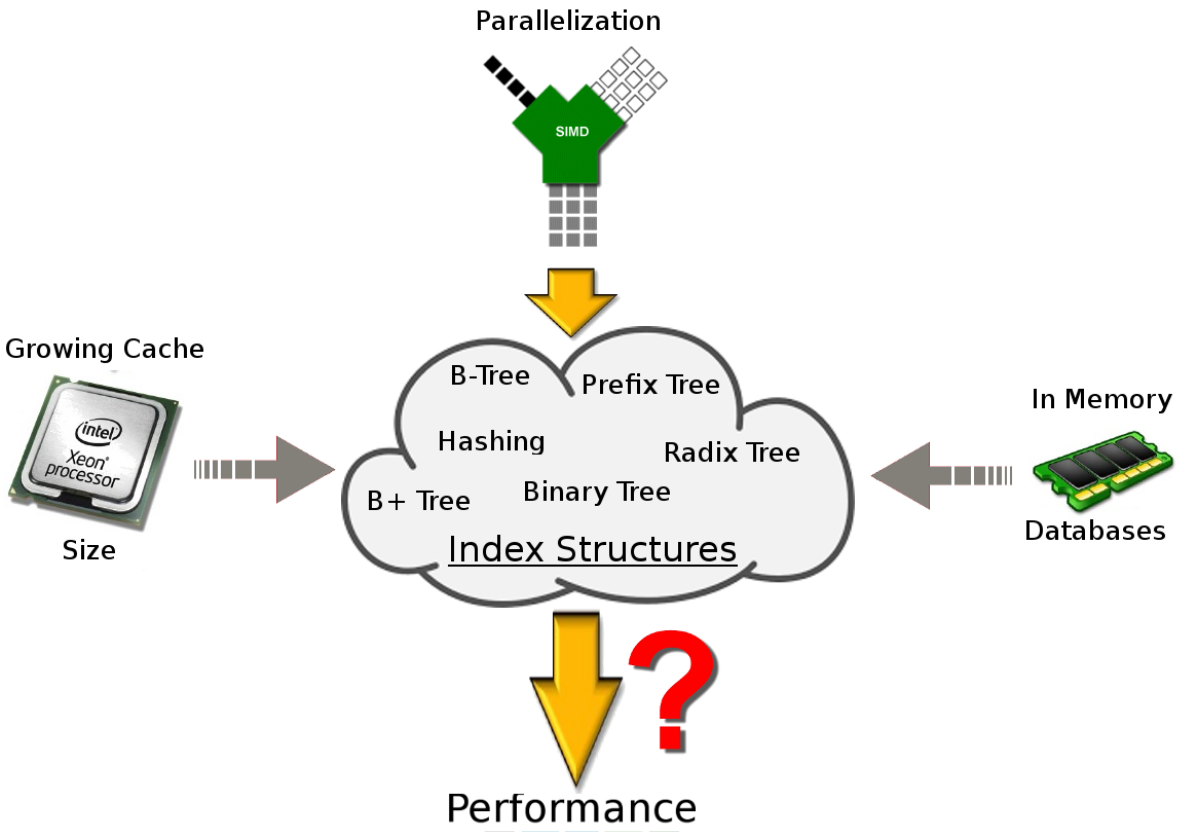
Adapted Index Structures

- Elf
- Seg-Tree/Trie
- FAST
- VAST
- ART

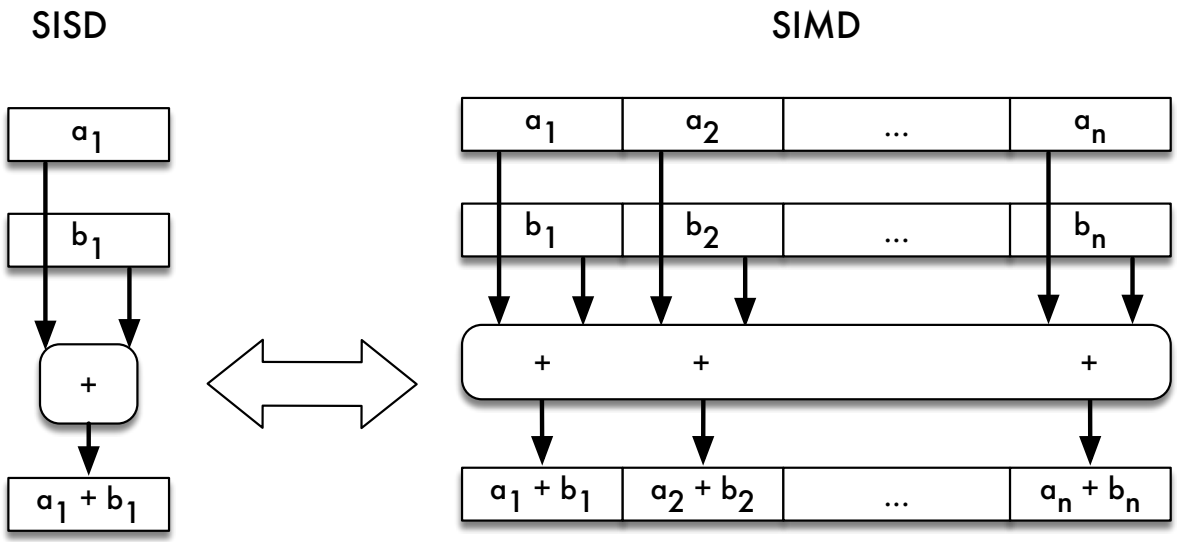
Evaluation

Conclusion

Motivation


Parallelization
SIMD
Growing Cache
intel
Xeon processor
Size
B-Tree
Prefix Tree
Hashing
Radix Tree
B+ Tree
Binary Tree
Index Structures
In Memory
Databases
?
Performance

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.

Horizontal Vectorization

- Using SIMD-instructions to compare data items in parallel
- Compare one search key to multiple keys of the index structure
  - Search key has to be duplicated
  - Often all keys of a node are compared against one search key

- Opposite: Vertical Vectorization
  - Not useful, since sequential search key storage in main memory is needed

Adapted Index Structures

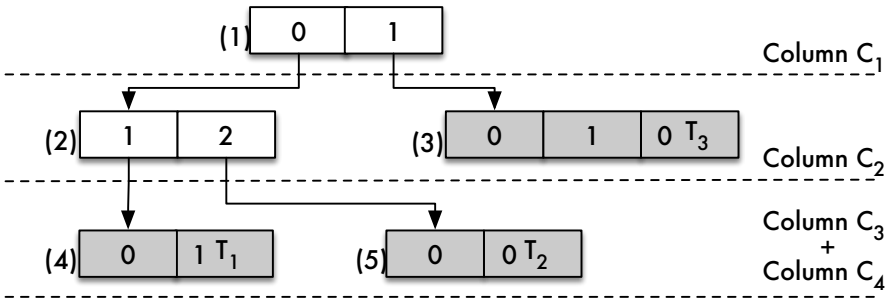
- Elf
- Seg-Tree/Trie
- FAST: Fast Architecture Sensitive Tree
- VAST: Vector-Advanced and Compressed Structure Tree
- ART: Adaptive Radix Tree

Elf

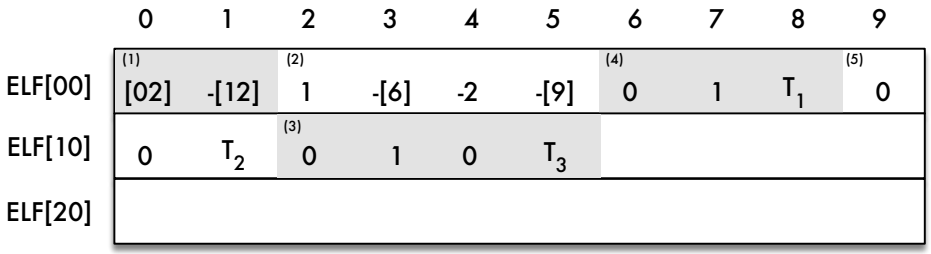
(a) Input Table

C_1	C_2	C_3	C_4	*Ref*
0	1	0	1	T_1
0	2	0	0	T_2
1	0	1	0	T_3

(b) Conceptual Elf

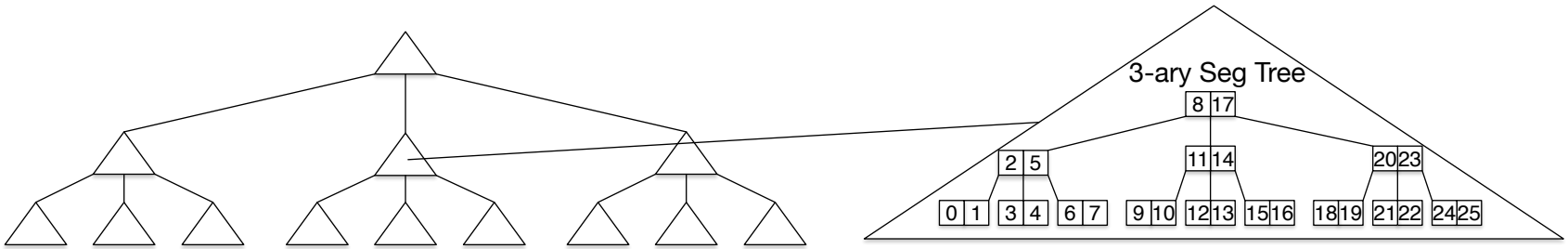


(c) Optimized Memory Layout for Elf



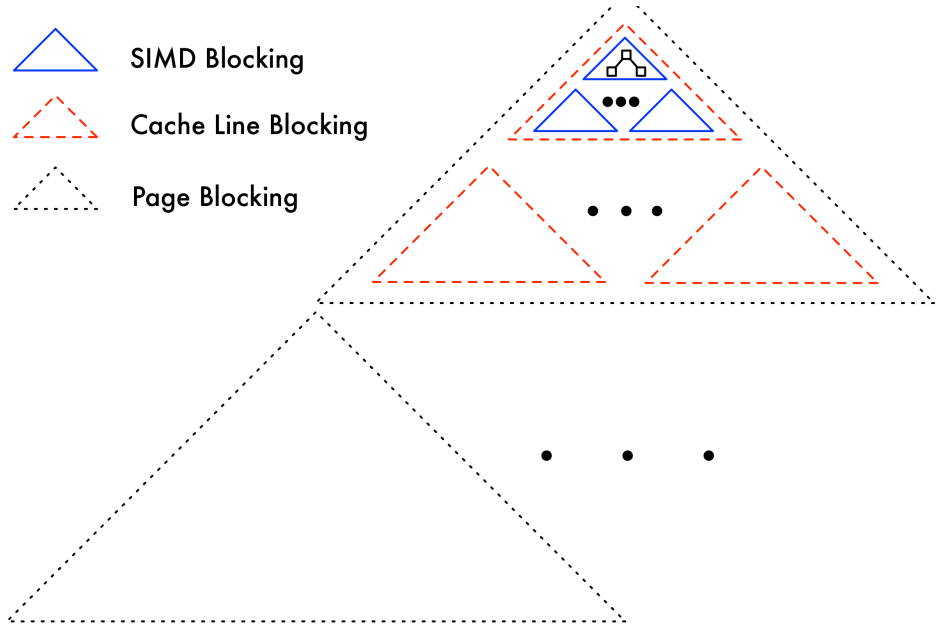
- Multi-dimensional index structure for column-wise storage
- Prefix-redundancy elimination on distinct column values
- Linearisation for optimized memory layout

Seg-Tree/Trie



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

Fast Architecture Sensitive Tree



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

Vector-Advanced and Compressed Structure Tree

- Extension of FAST
- Decrease branch misses avoiding conditional branches
- Uses key compression on lower levels of the tree
  - Lossy compression for inner nodes
  - Lossfree compression for child nodes
- Decompression and error correction of lossy compression has less impact compared to the performance increase with horizontal vectorisation

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
- Horizontal vectorization only on one node type
- Reduced space consumption due to lazy evaluation and path compression

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

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.

Thank you for your attention!