TECHNISCHE UNIVERSITÄT DRESDEN

Dresden Database Systems Group

Hardware-Oblivious SIMD Parallelism for In-Memory Column-Stores

Annett Ungethüm, Johannes Pietrzyk, Patrick Damme, Alexander Krause, Dirk Habich, Wolfgang Lehner

TU Dresden, Germany

Erich Focht

High Performance Computing Europe
NEC Deutschland GmbH, Germany

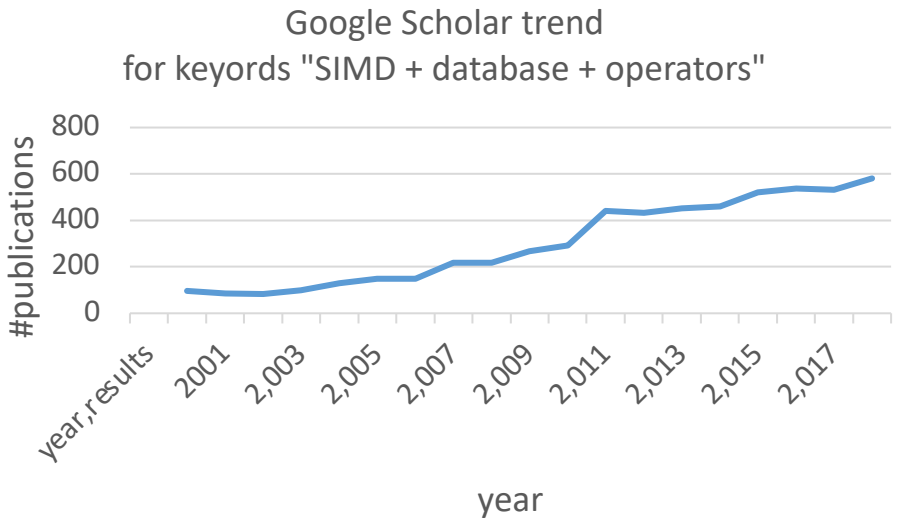
CIDR 2020, Amsterdam, Netherlands, 2020/01/14

Modern In-Memory Column-Stores

Dresden Database Systems Group

Single-Instruction Multiple Data (SIMD)

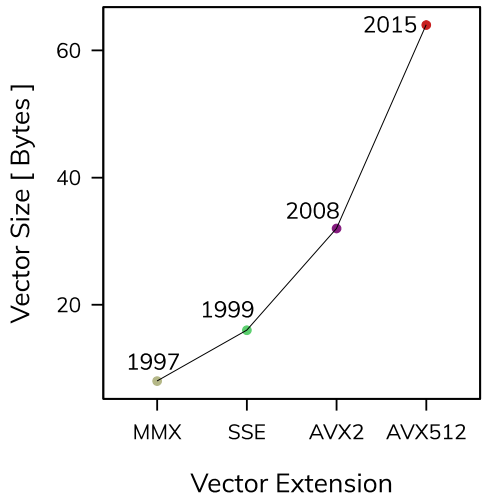
- State-of-the-art parallelism concept
- Execute same instruction on a set of values (vector unit, vector register) → vectorization
- Increases single-thread performance

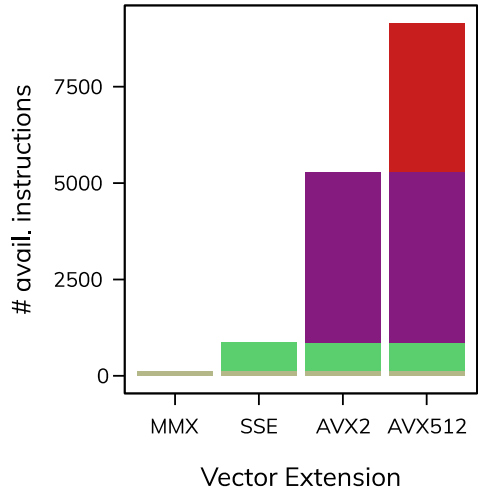


Common ground

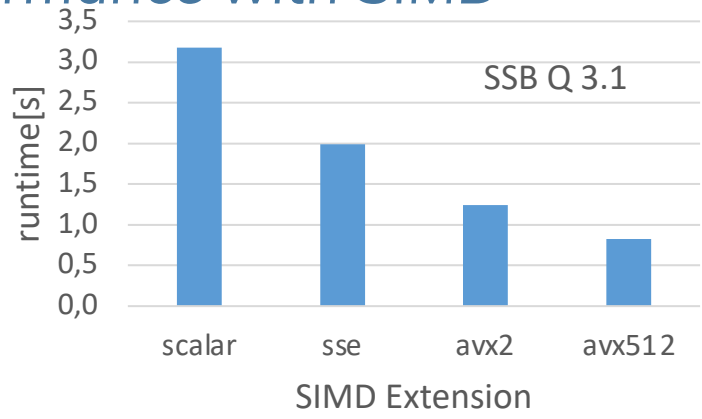
Explicit Vectorization using SIMD Intrinsics

Intel SIMD Extensions

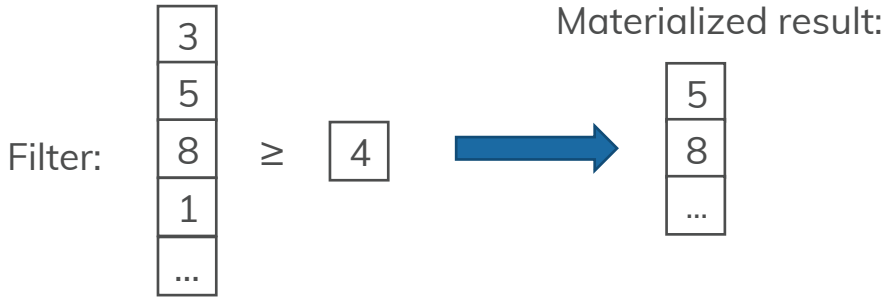




Performance with SIMD

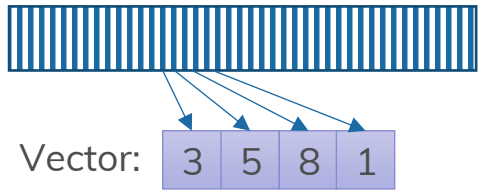


Explicit SIMD Programming: Running Example



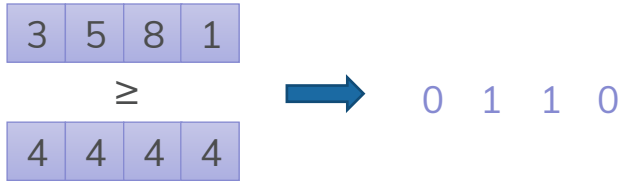
SSE

Load



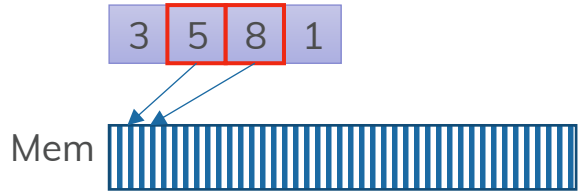

```
__m128i vec1 = _mm_load_si128(ptr_in);
```

Compare



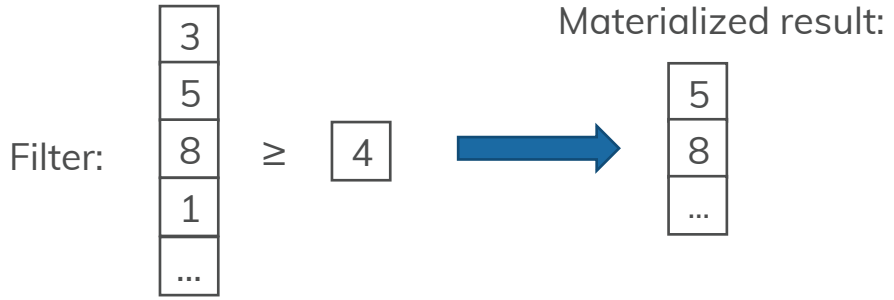

```
__m128i result = _mm_cmpgt_epi32(vec1, vec_const);
int mask = _mm_movemask_epi8(result);
```

Compress Store




```
switch (mask){
        case 0:   return;
        case 1:   *ptr_out = _mm_extract_epi32(p_vec,0);
                  return;
        ...
        case 12:  p_vec=_mm_shuffle_epi8(p_vec,
                    _mm_set_epi8(7,6,5,4,3,2,1,0,15,14,13,12,11,10,9,8));
                    _mm_storeu_si128(ptr_out, p_vec);
                    return;
        ...
        case 15:   _mm_storeu_si128(ptr_out, p_vec);
                  return;
    }
```

Porting to AVX512

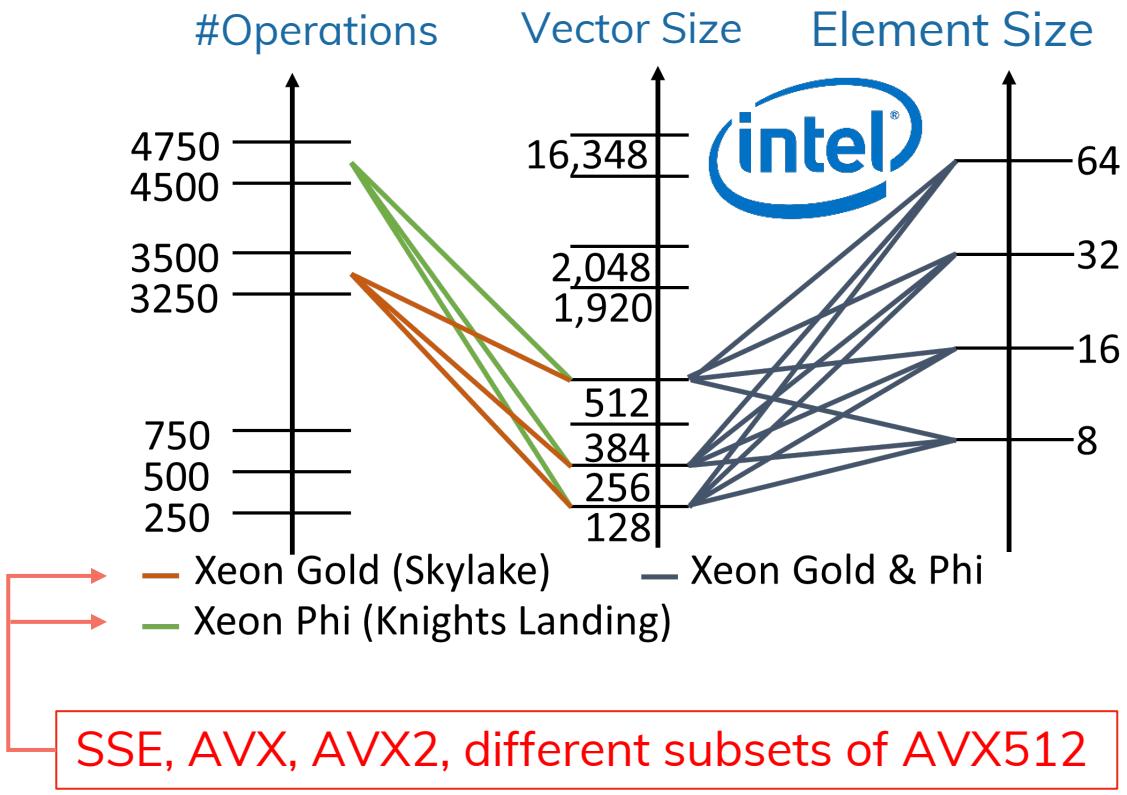


	SSE	AVX512
Load	__m128i vec1 = _mm_load_si128(ptr_in);	__m512i vec1 = _mm512_load_si512(ptr_in);
Compare	__m128i result = _mm_cmpgt_epi32(vec1, vec_const); int mask = _mm_movemask_epi8(result);	int mask = _mm512_cmpgt_epi32_mask(vec1, vec_const);

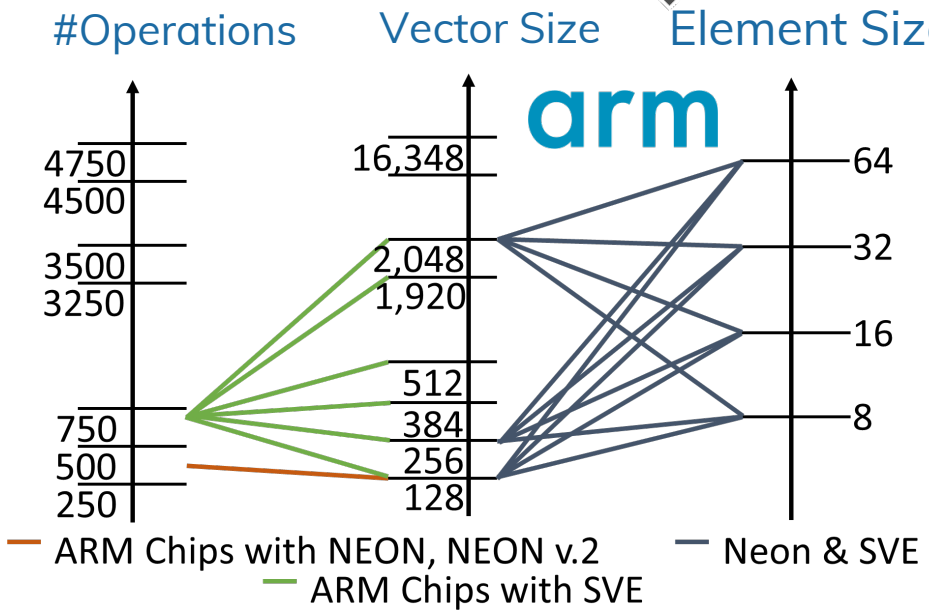
Compress Store	switch (mask){
	case 0: return;
	case 1: *ptr_out = _mm_
	return;
	...
	case 12: p_vec=_mm_shuffle_epi8(p_vec,
	_mm_set_epi8(7,6,5,4,3,2,1,0,15,14,13,12,11,10,9,8));
	_mm_storeu_si128(ptr_out, p_vec);
	return;
	...
	case 15: _mm_storeu_si128(ptr_out, p_vec);
	return;
	}
		_mm512_mask_compressstoreu_epi32(ptr_out, mask, vec1);

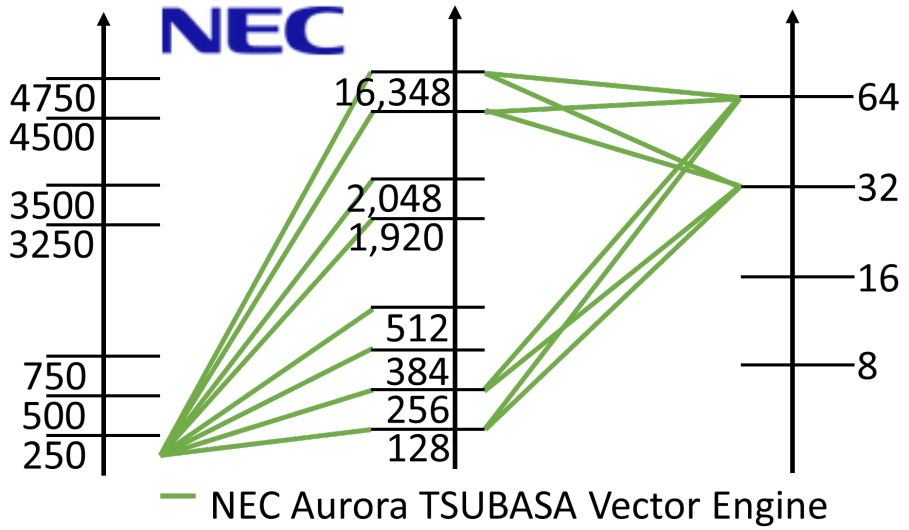
Major Challenge
An Abstraction to enable **Portablility** and **Extensibility**
But keep **explicit Vectorization**

SIMD Variety



SSE, AVX, AVX2, different subsets of AVX512

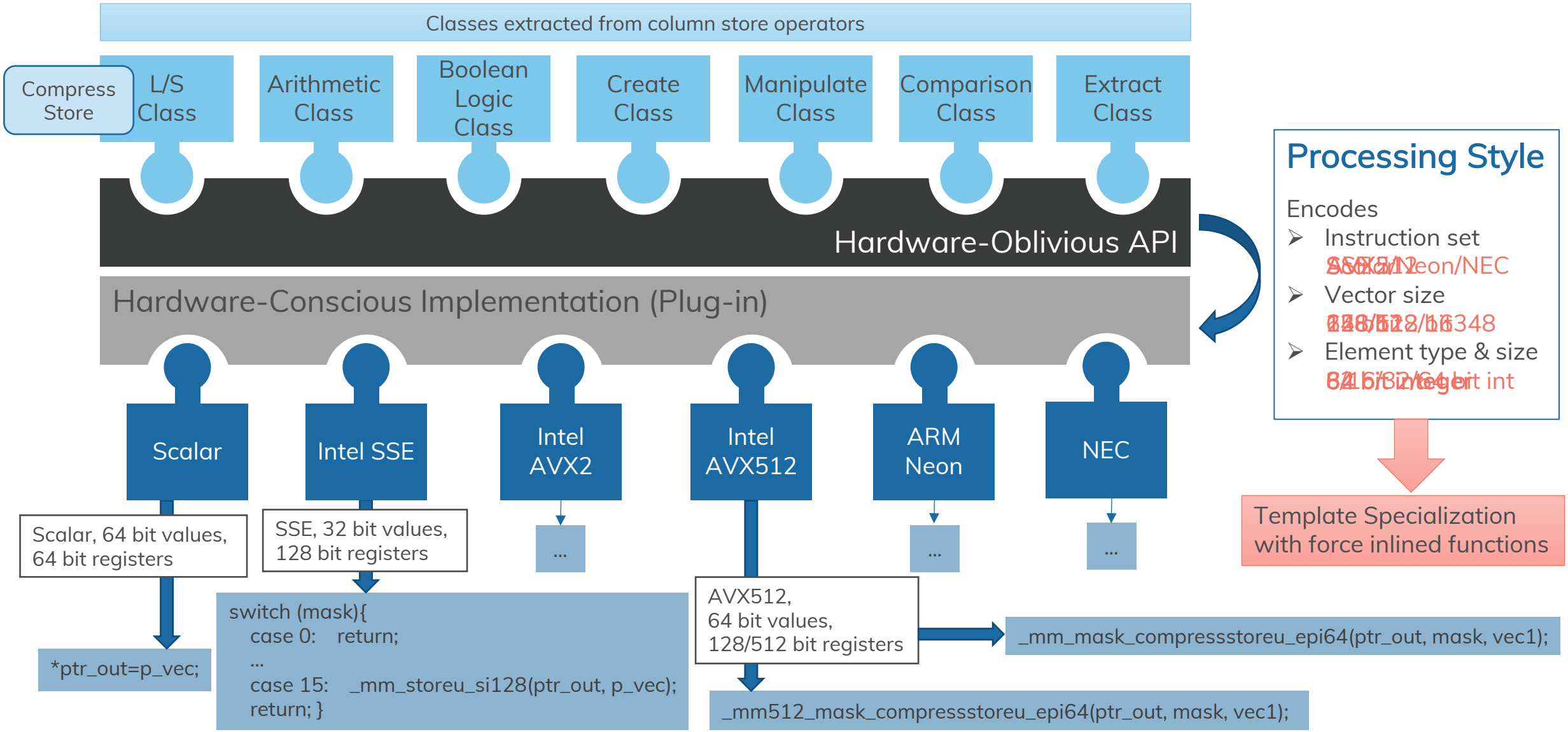




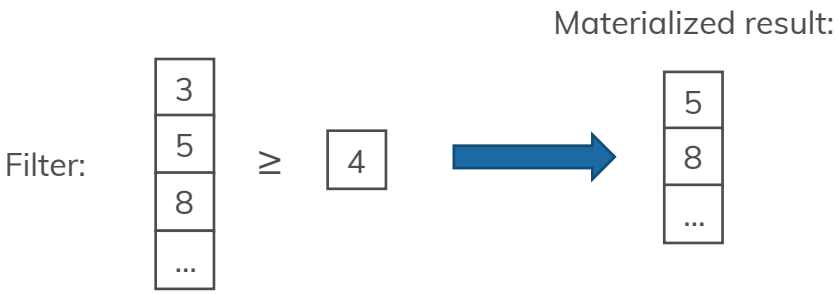
Porting across architectures is not trivial!

➢ Architecture independent API

Template Vector Library (TVL)



TVL in Practice



Dresden Database Systems Group

Primitives | TVL Data Types

Derieved Vector Properties

Instruction Set | Vector Size | Element Type

```
using processingStyle = avx512<512<32_t>>;
```

```
__m128i vec1 = _mm_load_si128(ptr_in);
```

```
vector_t vec1 = load <processingStyle, iov::ALIGNED, vector_size_bit>(ptr_in);
```

```
__m128i result = _mm_cmpgt_epi32(vec1, vec_const);
int mask = _mm_movemask_epi8(result);
```

```
mask_t mask = greater <processingStyle, vector_base_t_granularity>
                                                    (vec1, vec_const);
```

```
switch (mask){
        case 0:    return;
        case 1:    *ptr_out = _mm_extract_epi32(p_vec,0);
                   return;
        ...
        case 12:  p_vec=_mm_shuffle_epi8(p_vec,
                    _mm_set_epi8(7,6,5,4,3,2,1,0,15,14,13,12,11,10,9,8));
                    _mm_storeu_si128(ptr_out, p_vec);
                    return;
        ...
        case 15:    _mm_storeu_si128(ptr_out, p_vec);
                    return;
    }
```

SSE

Data alignment required for primitives in L/S class

```
compressstore <processingStyle, iov::UNALIGNED, vector_size_bit>
                                                    (ptr_out, vec1, mask);
```

TVL

End-to-End Evaluation

In-memory column store processing engine for analytical workloads

One code base for all operators

Backends for

- Intel SSE, AVX2, AVX512
- ARM Neon
- NEC SX-Aurora Tsubasa (partially)

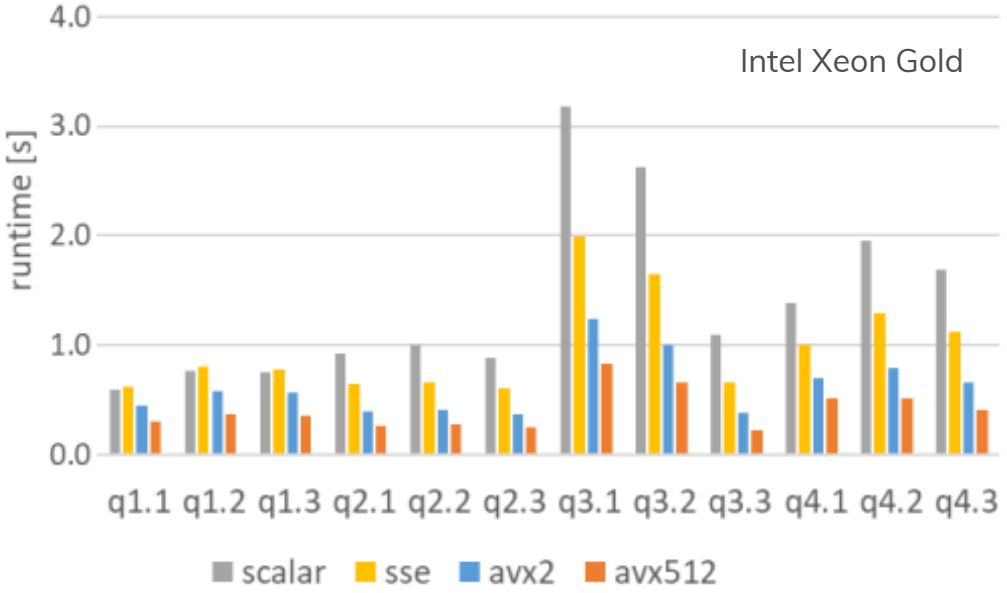
MorphStore

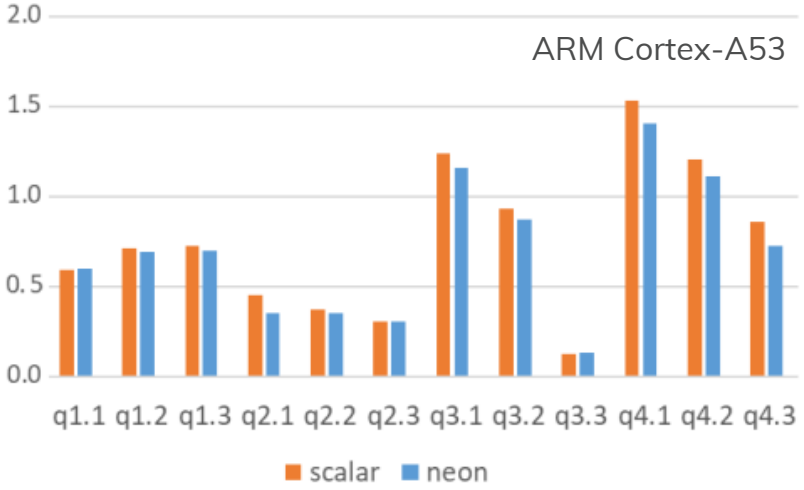
Research Prototype

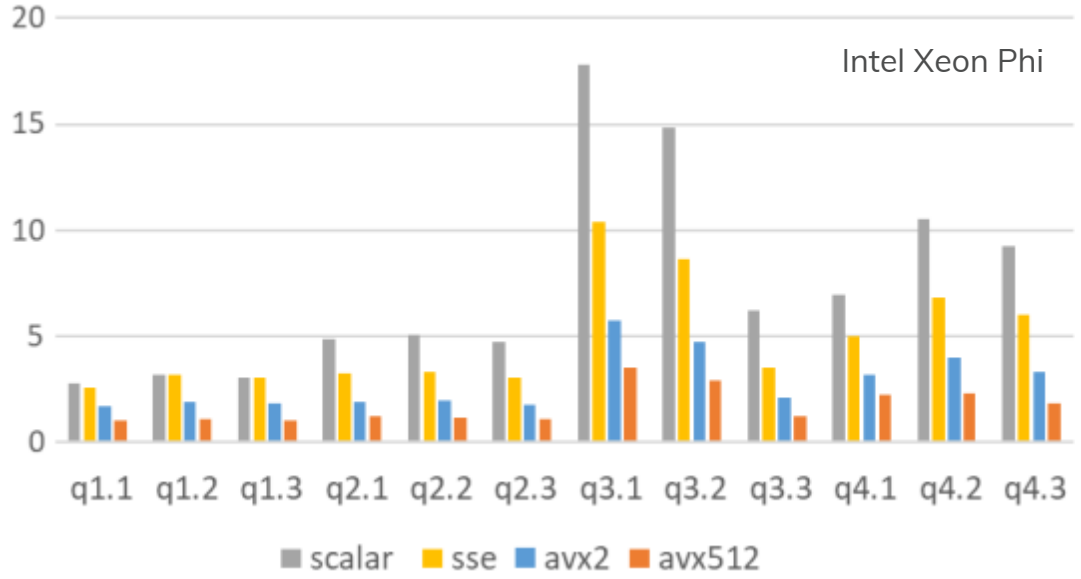
https://morphstore.github.io

D Habich, P Damme, A Ungethüm, J Pietrzyk, A Krause, J Hildebrandt, W Lehner "MorphStore-In-Memory Query Processing based on Morphing Compressed Intermediates LIVE." *Proceedings of the 2019 International Conference on Management of Data*. ACM, 2019.

Start Schema Benchmark

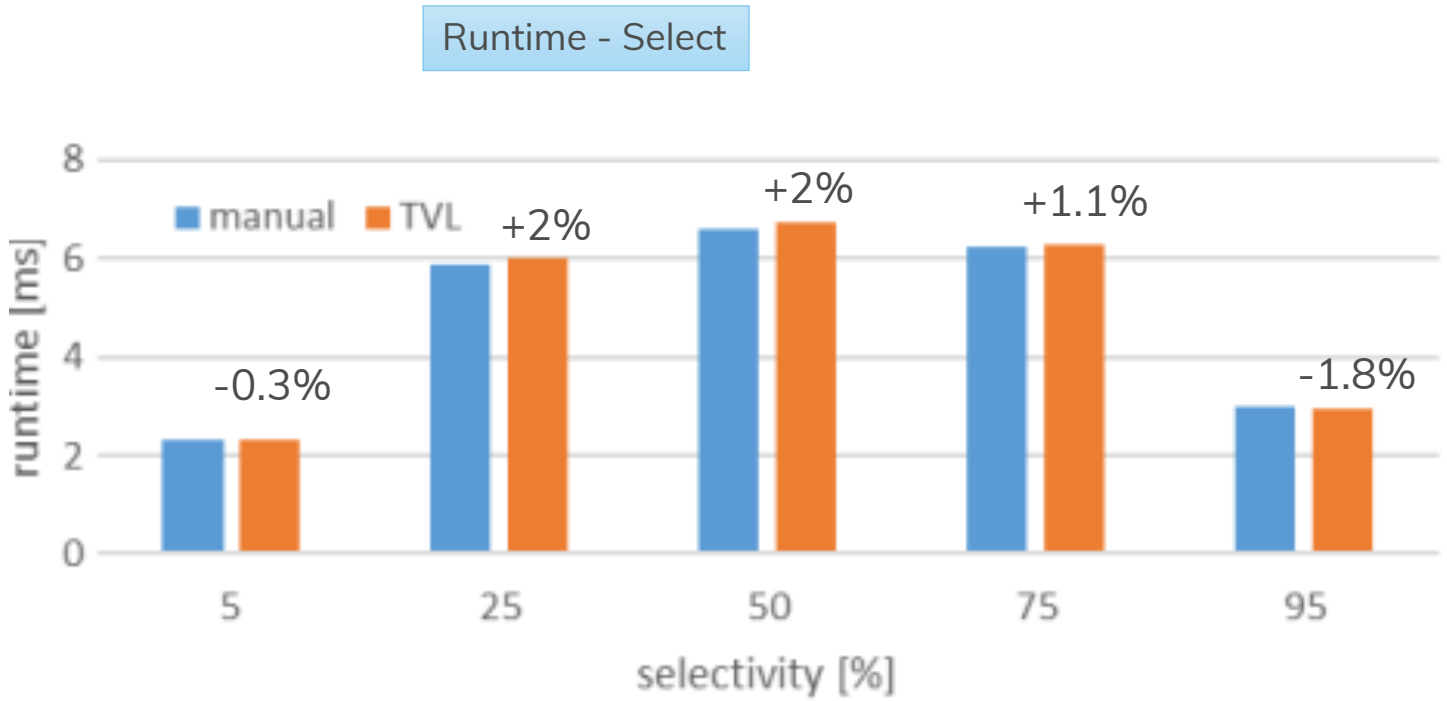






- *SSB runs on 5 different Instruction Sets and 4 different register sizes*
- *Only ONE codebase for all Benchmarks*
- *SIMD is benefitial in most cases*

Microbenchmarks for Runtime Overhead

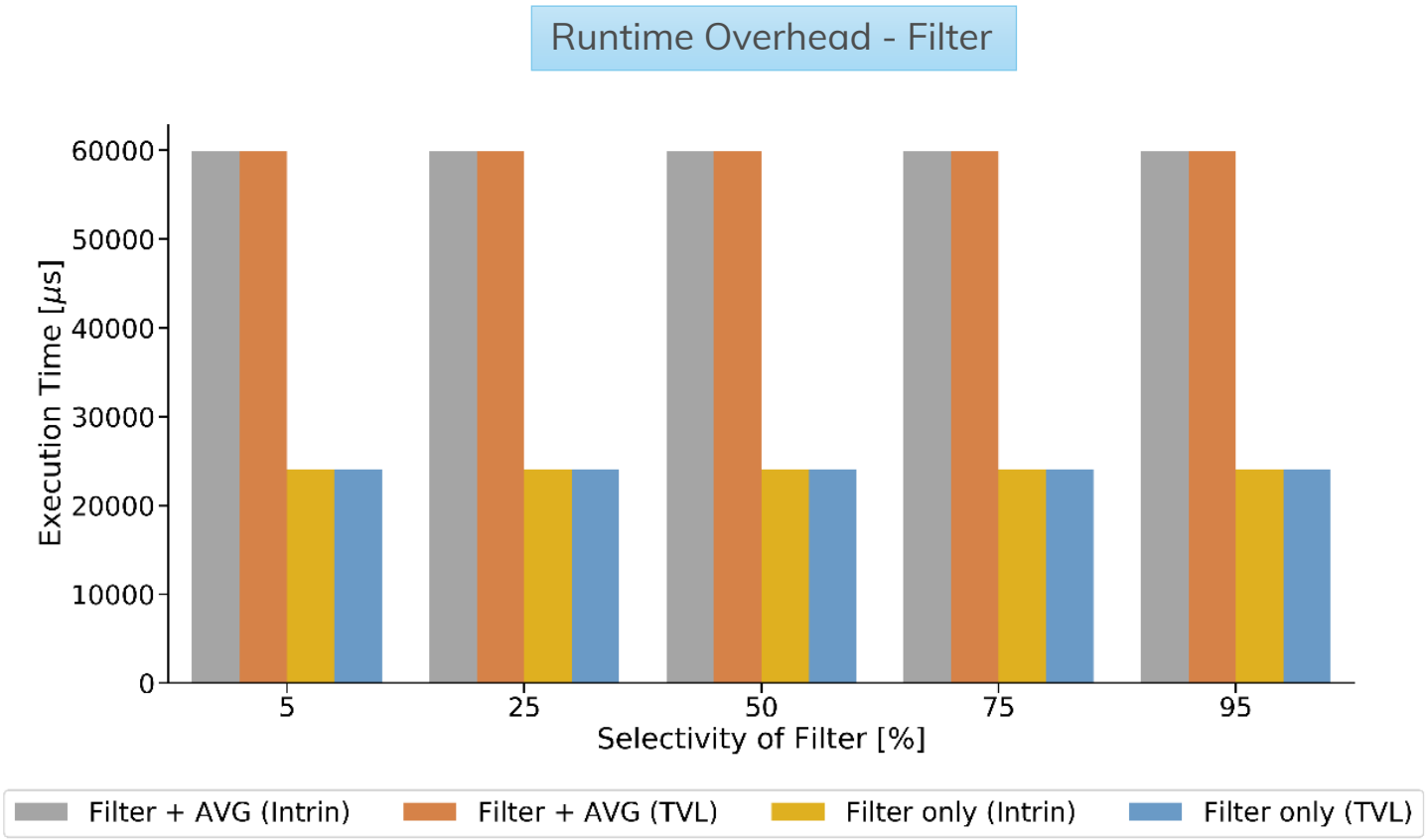


Test System: Intel Xeon Gold 5120

Comparison between TVL and hand-vectorized Operator (AVX2)

Virtually no overhead caused by TVL-library

Microbenchmarks for Runtime Overhead



Overhead of TVL over auto-vectorizer by NEC

TVL uses LLVM-VE backend providing intrinsics support

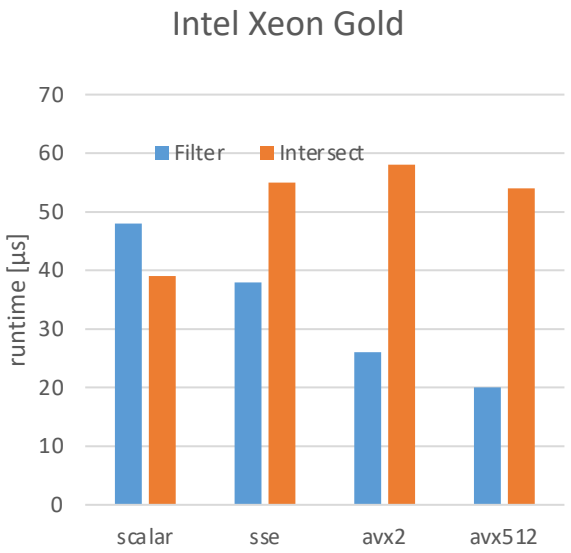
2 Cases:

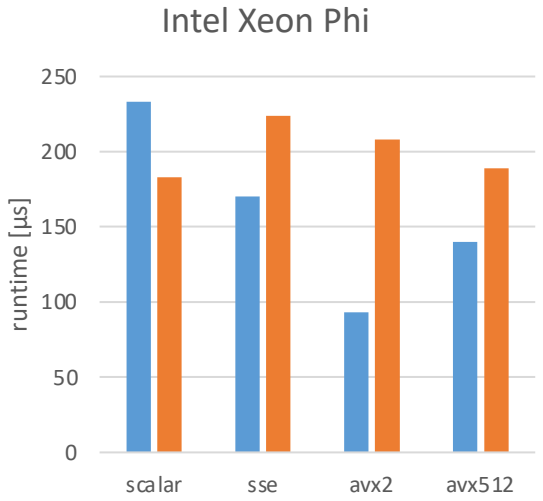
- Filter only
- Filter followed by aggregation

Virtually no overhead

Test System: NEC SX-Aurora Tsubasa

Microbenchmarks for Performance

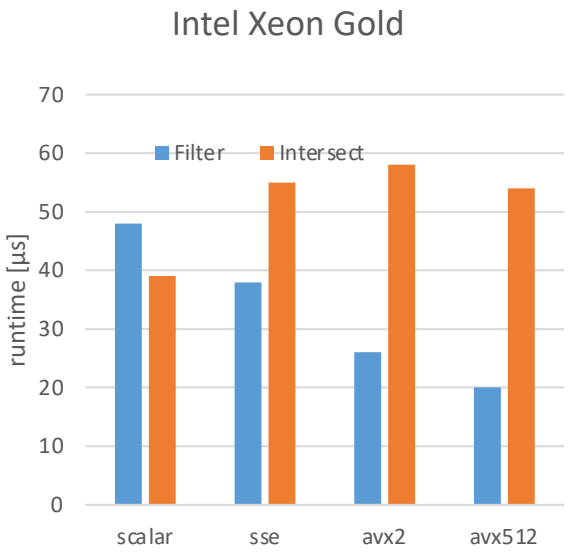


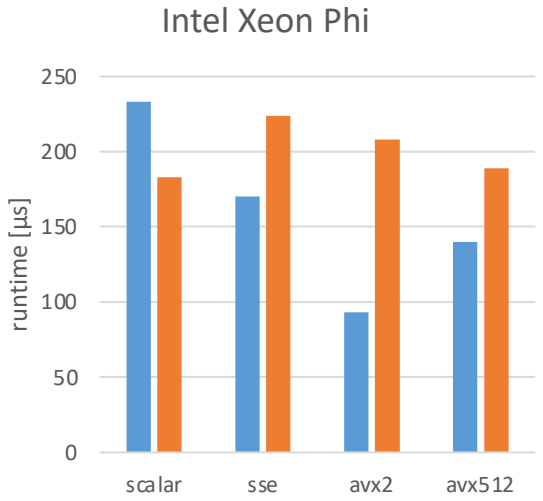


Same instruction set behaves differently on different systems

Longer vectors or newer instructions not always best choice

Future Work





Same instruction set behaves differently on different systems

Longer vectors or newer instructions not always best choice

➢ ***Exciting future work: Choice of Vector Extension as Optimization Knob***

TECHNISCHE UNIVERSITÄT DRESDEN

Dresden Database Systems Group

Hardware-Oblivious SIMD Parallelism for In-Memory Column-Stores

Annett Ungethüm, Johannes Pietrzyk, Patrick Damme, Alexander Krause, Dirk Habich, Wolfgang Lehner

TU Dresden, Germany

Erich Focht

High Performance Computing Europe
NEC Deutschland GmbH, Germany

CIDR 2020, Amsterdam, Netherlands, 2020/01/14