UA Rust Conference 2024

July 27 | online & offline | Learn. Develop. Discover.

All proceeds from the tickets will be donated to support **Ukraine**

near | Campus community | kumeka team | BOHEMIA AUTOMATION

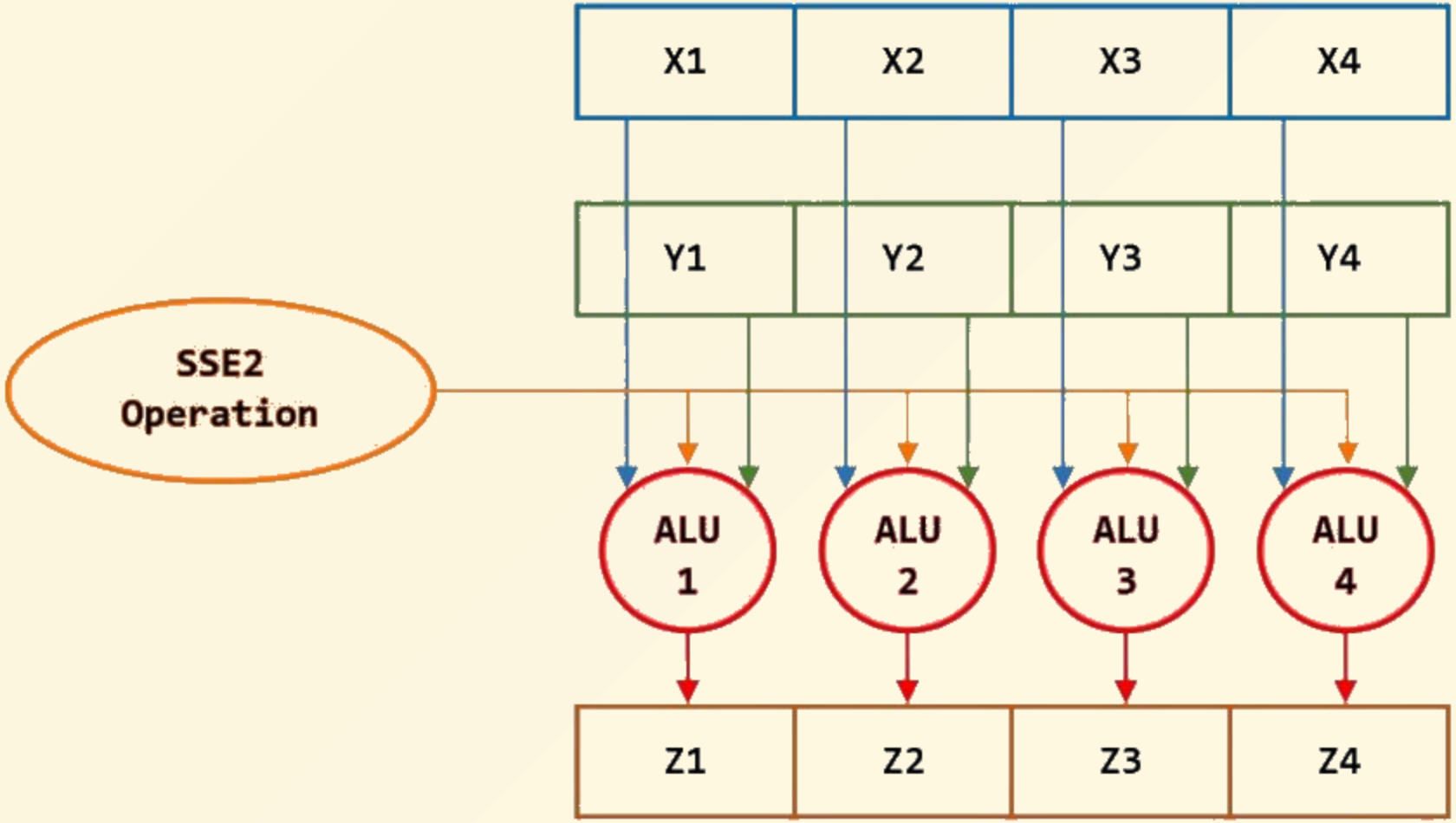
Processing texts with SIMD in Rust

Maxim Vorobjov
@
Volition Technologies

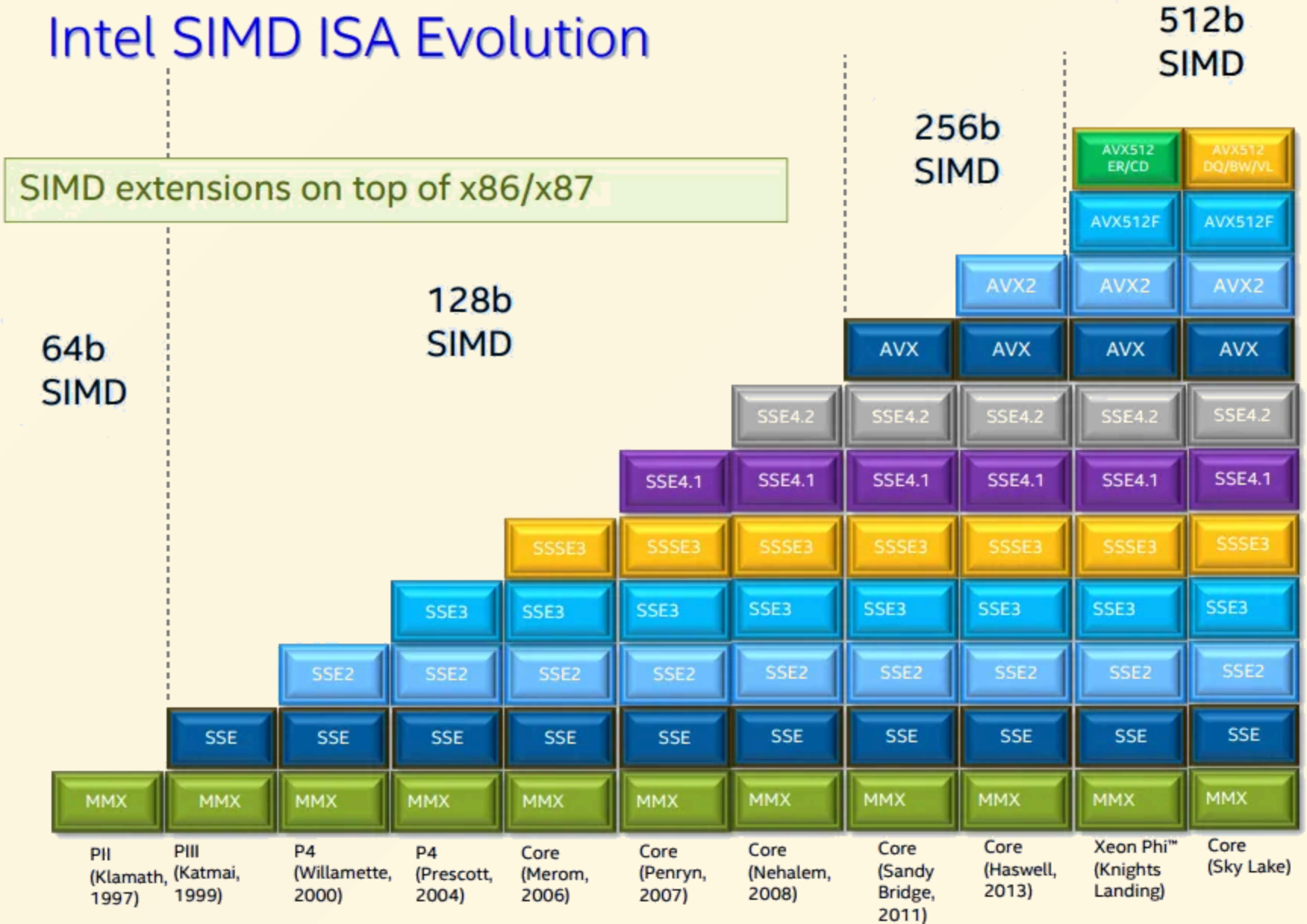
Plan

- What and why
- A few common use cases
- Examples presentation
- Disclaimer

Single Instruction Multiple Data



Intel SIMD ISA Evolution



Registers

	Size	Register
MMX	64-bit	XMM
SSE*	128-bit	XMM
AVX/2	256-bit	YMM
AVX512	512-bit	ZMM

“ AVX512 is disabled with 12th & 13th Gen ”

Support in Rust

stable

- LLVM is doing decent job vectorizing iterators
- `std::arch` provides access to SIMD primitives

> RUSTFLAGS='-C target-cpu=native'
> RUSTFLAGS='-C target-feature=+avx2'

nightly

- `portable_simd` feature provides type Simd<T, N>

When do we need it

- Processing large arrays of data
- But! Usually it is responsibility of compiler

In rare cases it might be used explicitly

- Language is not expressive enough for engineer to communicate abstractions
- Allows for faster processing and cost saving

> Danger! Beware of dragons and memory issues

Libraries

There are libraries that simplify using SIMD and allow you to make it cross-platform:

- memchr - allows for fast searching of characters and substrings in strings. Used by regex, ripgrep and other popular crates
- wide - vector types for cross-platform SIMD, similar to portable_simd on nightly but works with stable
- aho-corasick - library for fast searching of patterns in strings, also used by regex

👏 @ Ilya Zlobintsev

Text processing use cases

- Extract statistical information from text data
- Search/index large text documents
- Custom parsers to extract limited information

> For instance, extract one-two fields from the json

For example

- Count number of words
- Count number of times word is present in the text
- Extract few fields values from json

Count number of words in the text

Assume that the number of words is close to the number of spaces.
We will be only counting `' '` and `'\n'` characters.

For simplicity we assume that custom utf-8 characters are not used.

For huge texts small error in count is allowable.

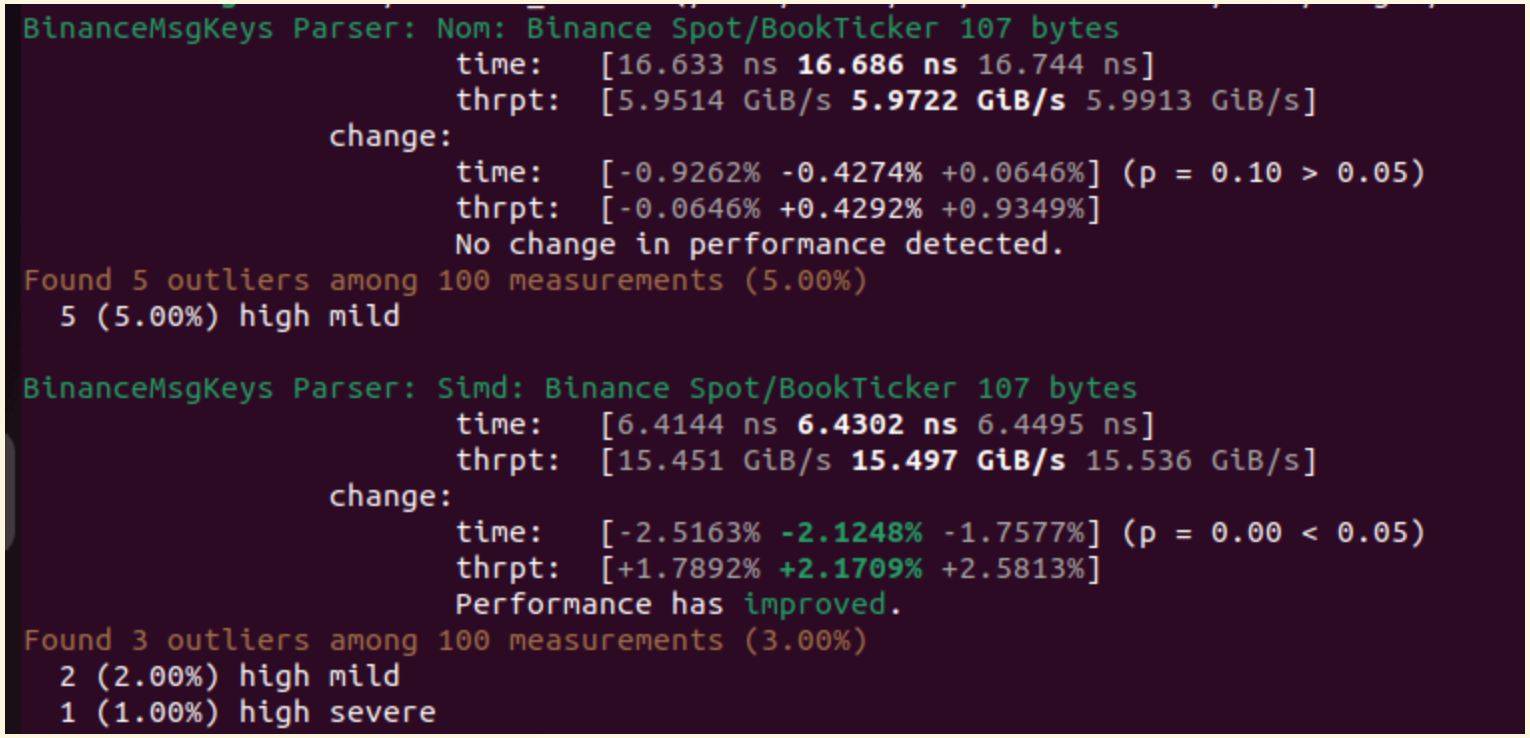
```
...
.LBB0_6:
        vmovd   xmm5, dword ptr [rdi + rax]
        vmovd   xmm6, dword ptr [rdi + rax + 4]
        vmovd   xmm7, dword ptr [rdi + rax + 8]
        vmovd   xmm8, dword ptr [rdi + rax + 12]
        vpcmpeqb        k0, xmm5, xmm1
        vpcmpeqb        k1, xmm6, xmm1
        vpcmpeqb        k2, xmm7, xmm1
        vpcmpeqb        k3, xmm8, xmm1
        vpmovm2q        ymm5, k0
        vpsubq  ymm0, ymm0, ymm5
        vpmovm2q        ymm5, k1
        vpsubq  ymm2, ymm2, ymm5
        vpmovm2q        ymm5, k2
        vpsubq  ymm3, ymm3, ymm5
        vpmovm2q        ymm5, k3
        vpsubq  ymm4, ymm4, ymm5
...
```

Count number of times word is found in the text

We will find matching substring with exact match to input query.

Extract 2 fields from small json (107b) message from Binance exchange

(no implementation provided)



Disclaimer

This approach can only be used in exceptional cases

- Explicit usage of SIMD provides much less readable code
- Code is using unsafe and requires extra care
- LLVM with cpu-target native provides decent vectorization

But

- It might improve performance / reduce costs 2-20 times

Thank you

Q & A