Compression Carcinized

implementing zlib-rs

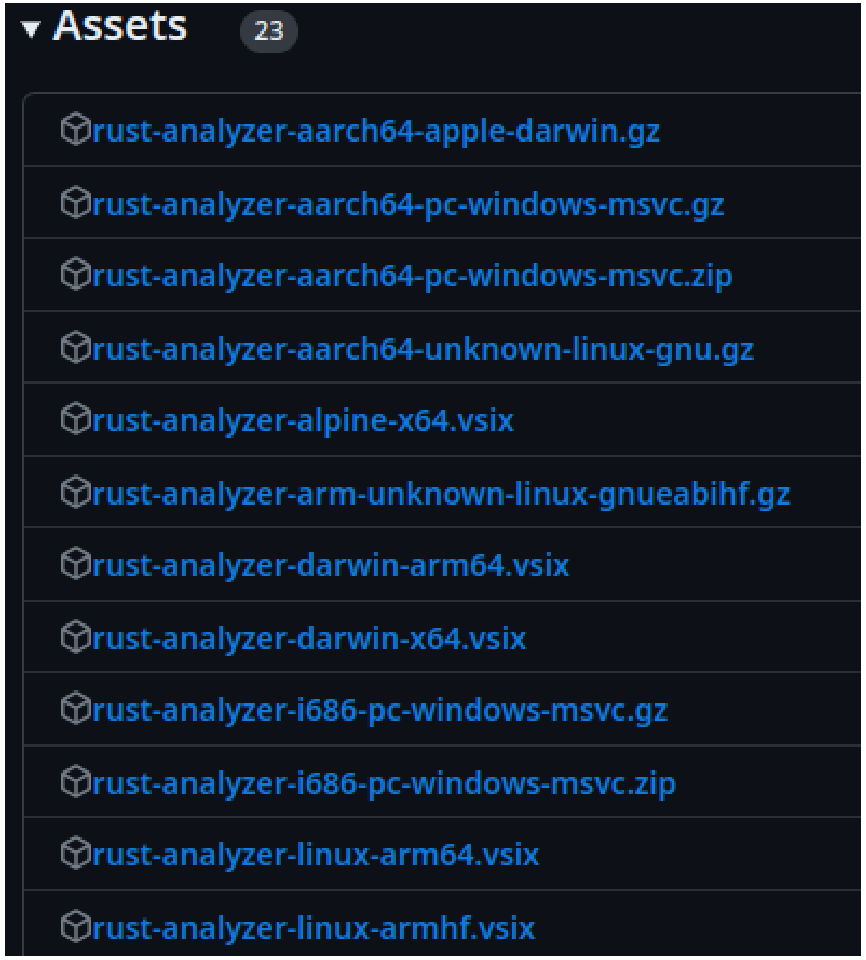
Folkert de Vries, RustNL 2024

carcinization

🦞 ➡️ 🦀

a form of convergent evolution in which non-crab crustaceans evolve a **crab-like body plan**

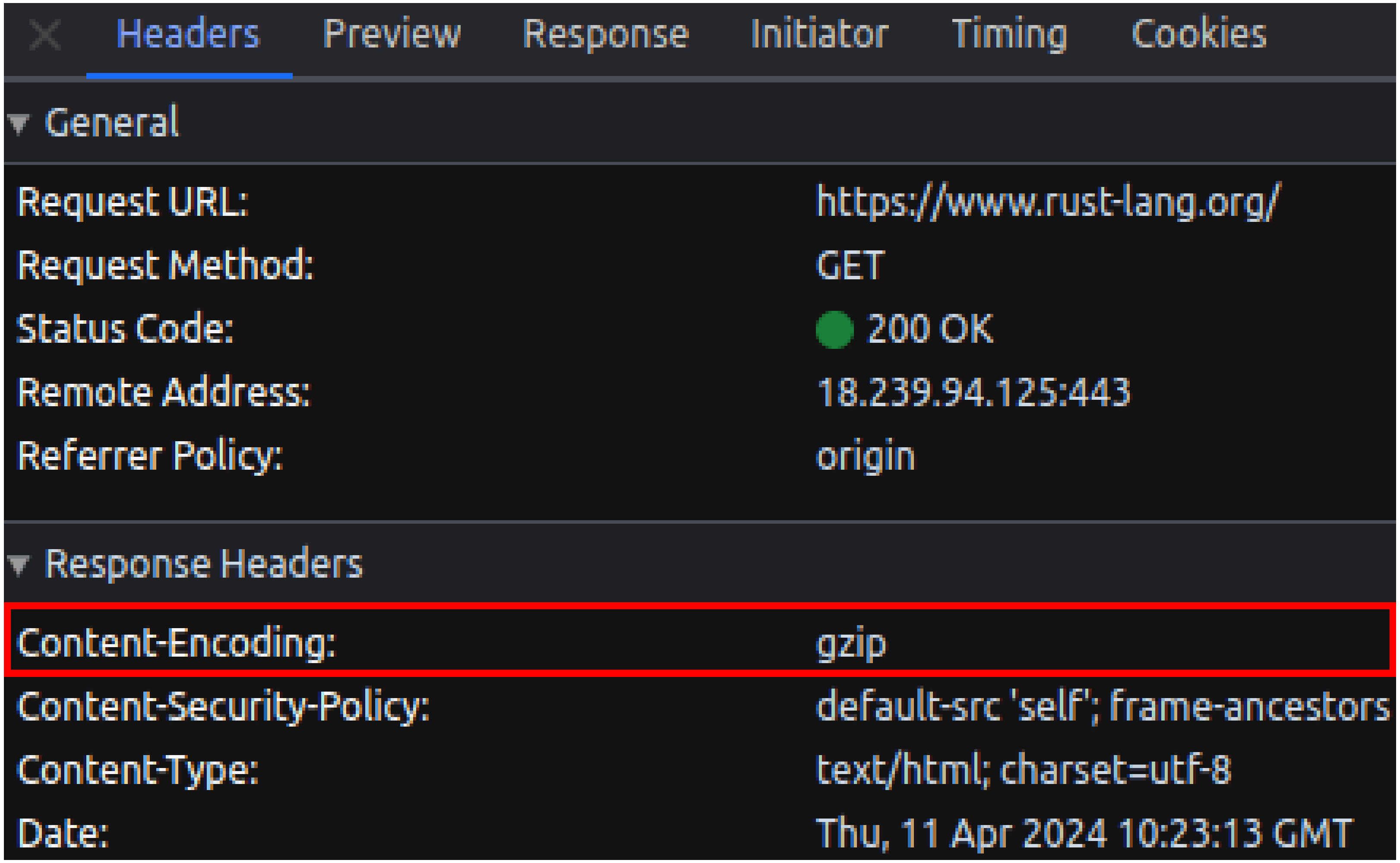
zlib: a library you've used today



▾ Assets 23

rust-analyzer-aarch64-apple-darwin.gz
rust-analyzer-aarch64-pc-windows-msvc.gz
rust-analyzer-aarch64-pc-windows-msvc.zip
rust-analyzer-aarch64-unknown-linux-gnu.gz
rust-analyzer-alpine-x64.vsix
rust-analyzer-arm-unknown-linux-gnueabihf.gz
rust-analyzer-darwin-arm64.vsix
rust-analyzer-darwin-x64.vsix
rust-analyzer-i686-pc-windows-msvc.gz
rust-analyzer-i686-pc-windows-msvc.zip
rust-analyzer-linux-arm64.vsix
rust-analyzer-linux-armhf.vsix

zlib: a library you've used today



▼ General

Request URL:	https://www.rust-lang.org/
Request Method:	GET
Status Code:	● 200 OK
Remote Address:	18.239.94.125:443
Referrer Policy:	origin

▼ Response Headers

Content-Encoding:	gzip
Content-Security-Policy:	default-src 'self'; frame-ancestors
Content-Type:	text/html; charset=utf-8
Date:	Thu, 11 Apr 2024 10:23:13 GMT

zlib: a library you've used today

```
> objdump -T /usr/lib/x86_64-linux-gnu/libz.so | grep "compress"
0000000000010370 g    DF .text  0000000000000022  ZLIB_1.2.0  compressBound
0000000000010360 g    DF .text  000000000000000f  Base        compress
0000000000010220 g    DF .text  000000000000013d  Base        compress2
0000000000010560 g    DF .text  000000000000001c  Base        uncompress
00000000000103a0 g    DF .text  00000000000001c0  ZLIB_1.2.9  uncompress2
```

```
pub unsafe extern "C" fn compress2(
    dest: *mut Bytef,
    destLen: *mut c_ulong,
    source: *const Bytef,
    sourceLen: c_ulong,
    level: c_int
) -> c_int
```

project goals

📚

drop-in replacement for the zlib dynamic library

📦

high-performance implementation for rust

Topics

- crash course compression
- the zlib ecosystem
- ponderings on porting

Why use **many** byte when **few** do trick?

Why Compress?

cost

speed

Lossless Compression

```
assert_eq!(decompress(compress(data)), data)
```

Recognizing patterns

foobarfoo

⬇️

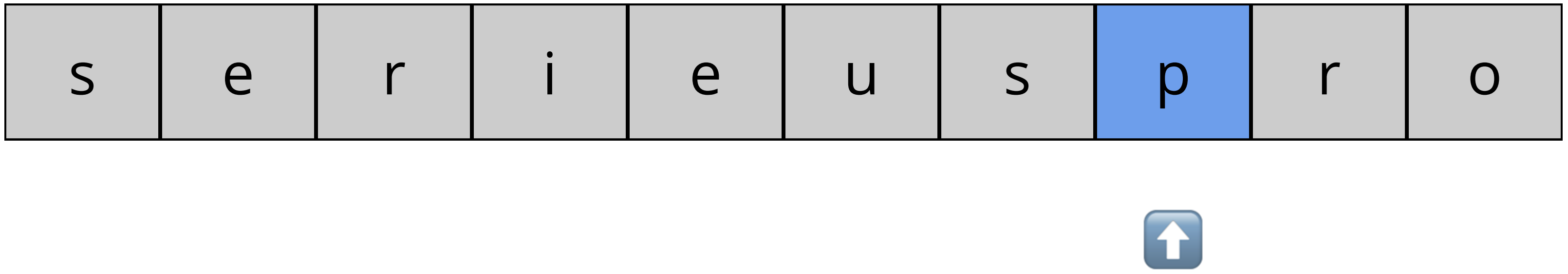
foobar<offset = 6, len = 3>

Finding patterns

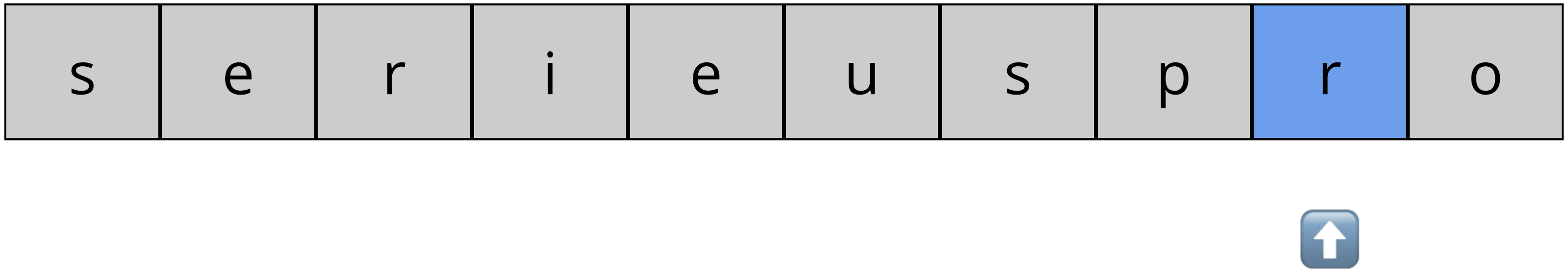
3.14159265358979323846264338327950288419716939937510582097494459230781640628620899862803482534211706798214808651328230664709384460955058223176

goal: find the (longest) <offset,len> insertions

Finding patterns

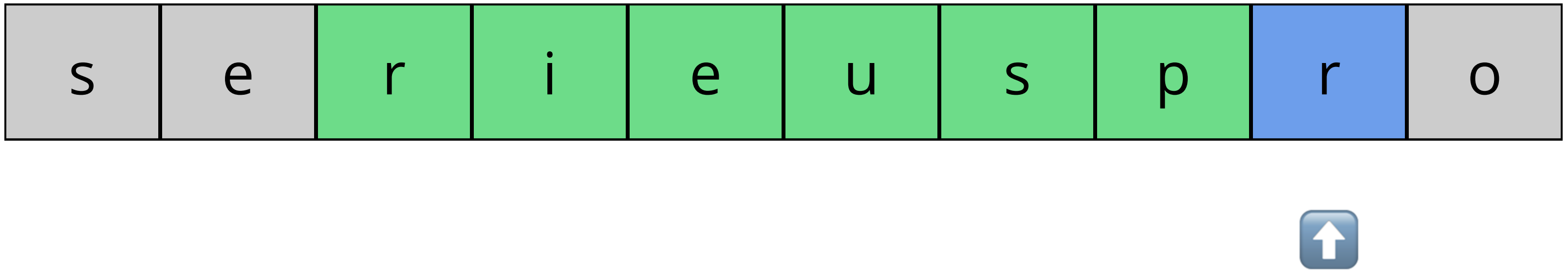


Finding patterns



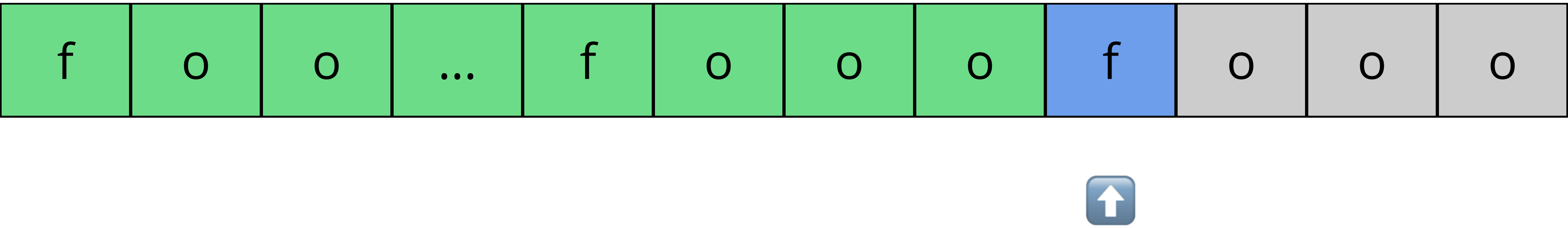
Finding patterns

The **window size** determines how far back the offset can go

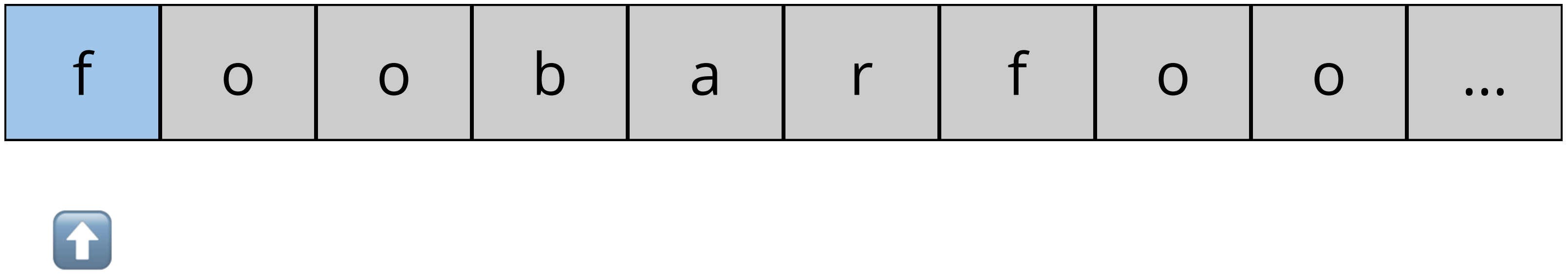


Finding patterns

The **compression level** determines how hard we try to find the longest match

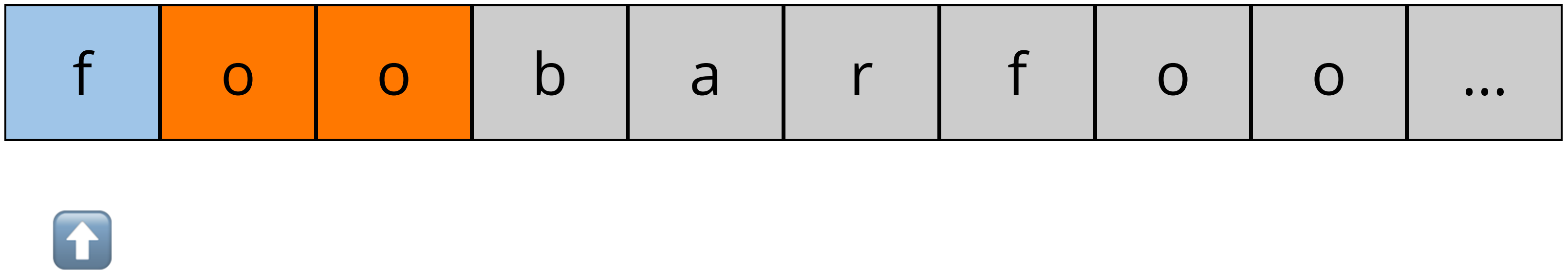


Finding patterns



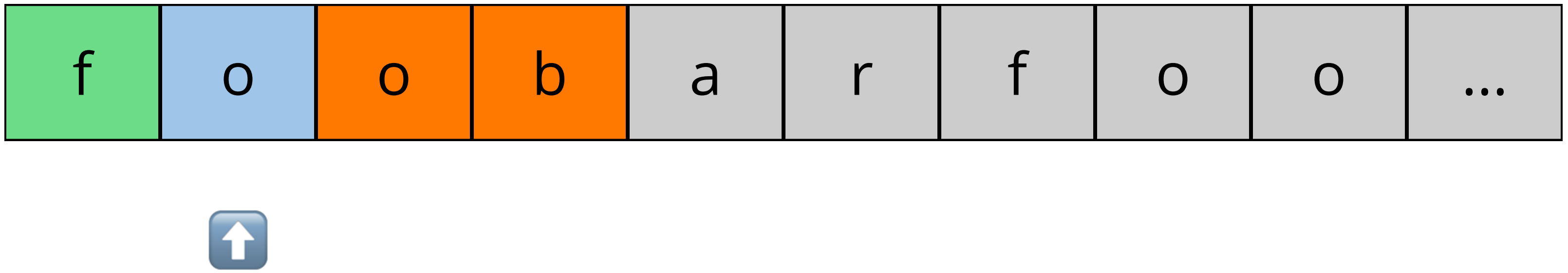
Finding patterns

```
"foo" -> { 0 }
```



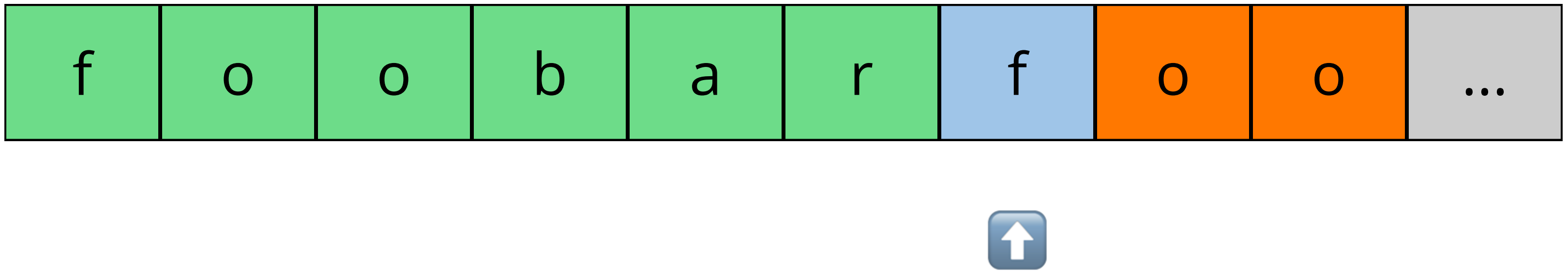
Finding patterns

```
"foo" -> { 0 }
"oob" -> { 1 }
```



Finding patterns

```
"foo"  -> { 0 }
"oob"  -> { 1 }
"oba"  -> { 2 }
"bar"  -> { 3 }
"arf"  -> { 4 }
"rfo"  -> { 5 }
```



Finding patterns

🌐

very effective for web data, even at low compression levels

Streaming

zlib can stream compression and decompression

zlib & rust

what are we even implementing

zlib-adler: the OG

goal: **stability**

still supports 16-bit systems

does not use modern hardware well

zlib-ng: the next generation

goal: **performance**

removes legacy,
but API-compatible

uses SIMD to speed up
the algorithm

why rust

reduced surface area

why rust

“ Any sufficiently complicated C project contains an

ad hoc,

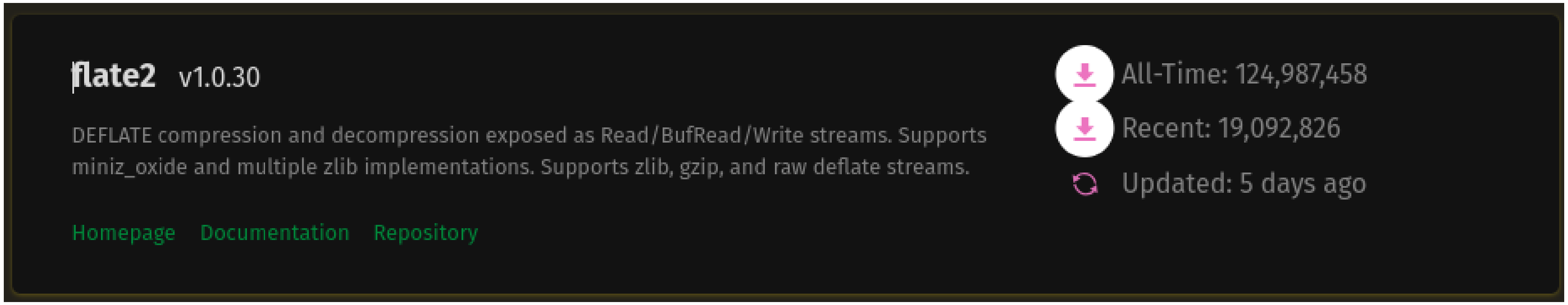
informally-specified,

bug-ridden,

slow

implementation of half of cargo

flate2



flate2 v1.0.30

DEFLATE compression and decompression exposed as Read/BufRead/Write streams. Supports miniz_oxide and multiple zlib implementations. Supports zlib, gzip, and raw deflate streams.

Homepage Documentation Repository

All-Time: 124,987,458

Recent: 19,092,826

Updated: 5 days ago

a nice rust API for zlib

used in cargo

miniz-oxide: better safe than sorry

goal: **safety**

a safe (but slow) rust implementation

does not cover the full zlib API

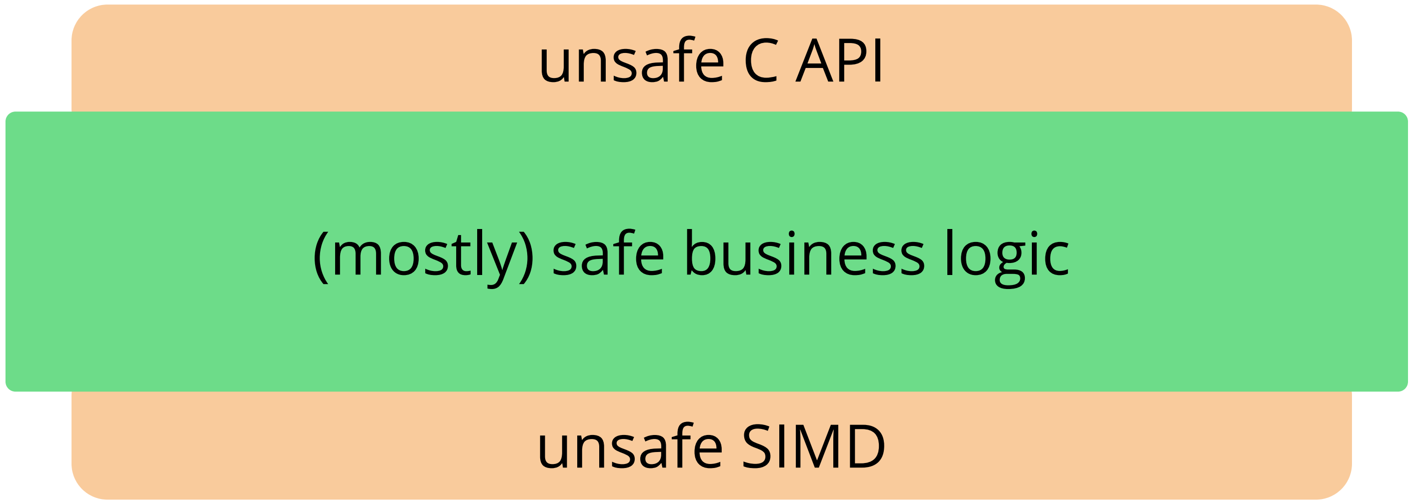
zlib-rs: a safer zlib

goal: **safety & performance**

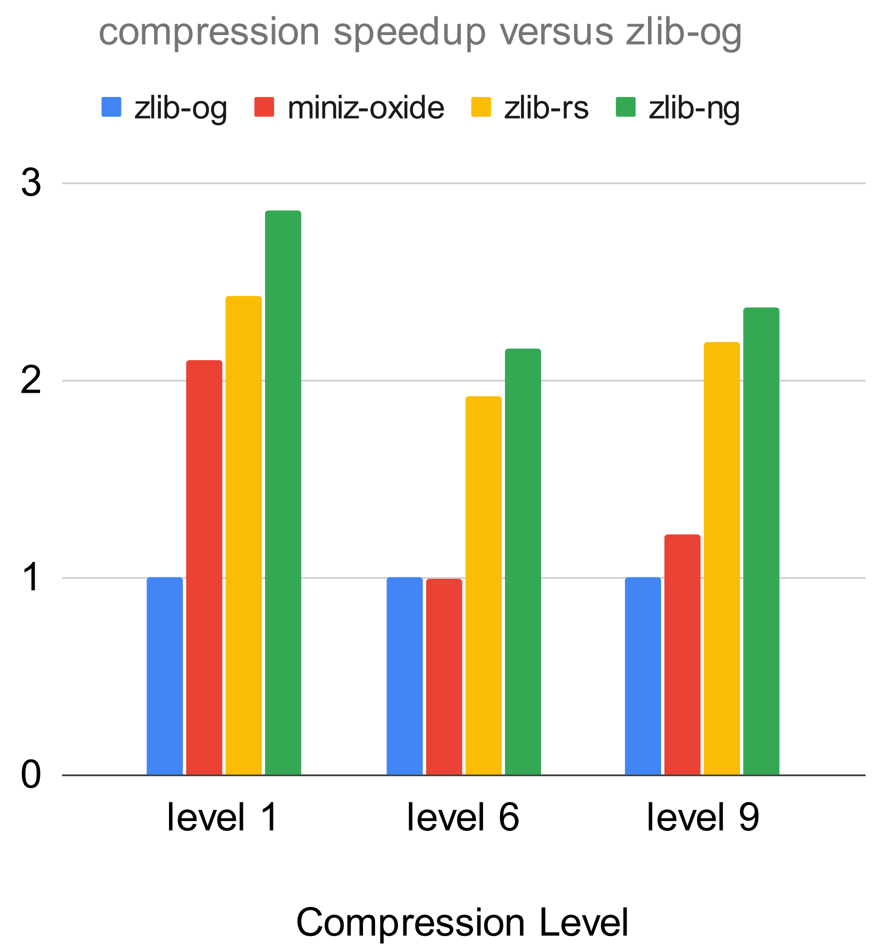
faster through the use of SIMD

implements the full zlib API

unsafe sandwich



compression speed



progress

```toml
[dependencies]
flate2 = {
    version = "1.0.29",
    default-features = false,
    features = ["zlib-rs"]
}
```

early days, but give it a go!

a crab-like body plan

🌊 ➡️ 🦀

from C to rust

spectrum of porting

🌳 implementation rewrite 🏭

spectrum of porting

🌳 implementation rewrite 🏭

⬆️

implementation

- e.g. rustls, ntpd-rs
- reinvent the wheel
- innovate on architecture
- high risk, high reward

spectrum of porting

🌳 implementation rewrite 🏭

⬆️

implementation

- e.g. rustls, ntpd-rs
- reinvent the wheel
- innovate on architecture
- high risk, high reward

Rewrite

- e.g. rav1d
- use existing knowledge
- inherits architecture
- works on day 1

spectrum of porting

🌳 🏭

implementation **rewrite**

```
// ...
i = 0 as libc::c_int;
while i < nblock {
    ftab[*eclass8.offset(i as isize) as usize] += 1;
    ftab[*eclass8.offset(i as isize) as usize];
    i += 1;
    i;
}
// ...
```

spectrum of porting

🌳 implementation rewrite 🏭

⬆️

zlib

- reuse existing knowledge (correctness & performance)
- quick results
- architecture is constrained anyway

RiiR and You

compatability

🧩

just better

🚀

RiiR and You

funding

💰

adoption

👥🤖

Summary

why use **many** bytes when **few** do trick

unreasonably effective on web content

try **zlib-rs**

use more (unglamorous) rust **in production**

🦀evolve crab-like body plans🦀

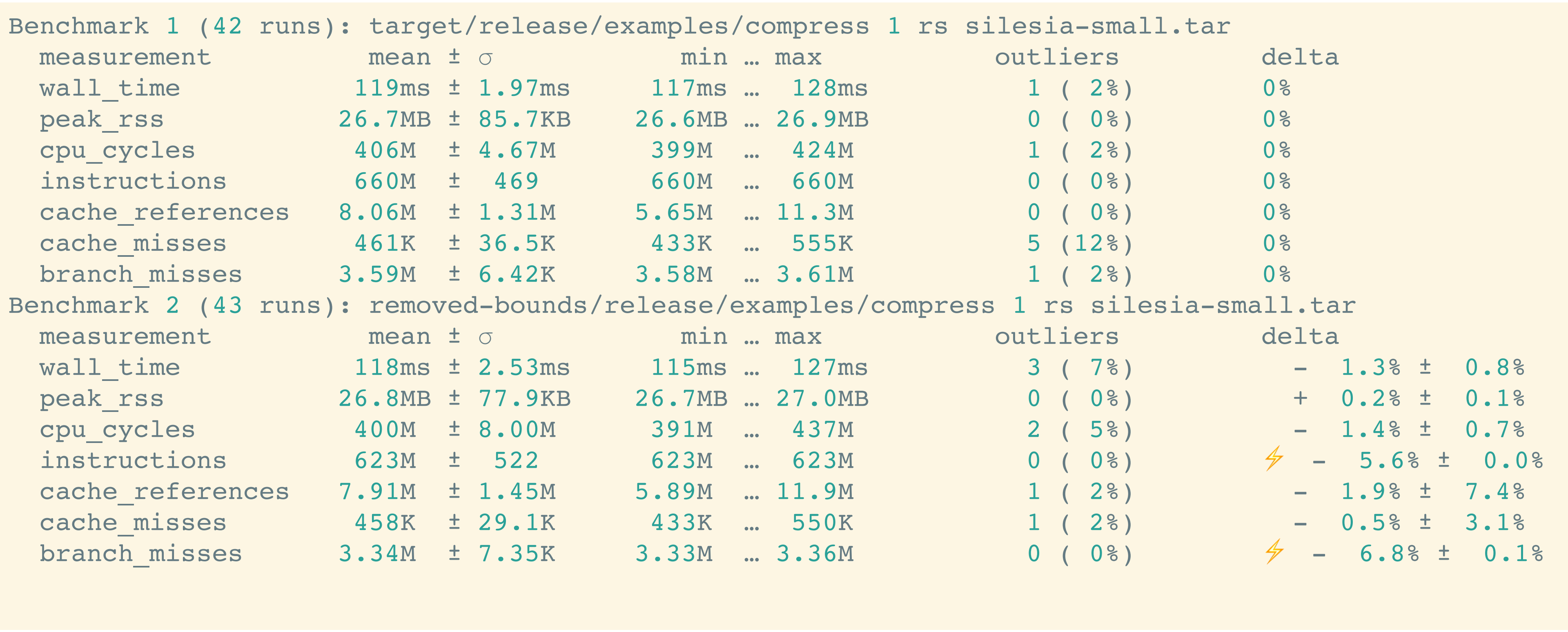
Thanks

github.com/memorysafety/zlib-rs

memorysafety.org/initiative/zlib/

slides.com/folkertdevries/compression-carcinized

is it bounds checks?



```
Benchmark 1 (42 runs): target/release/examples/compress 1 rs silesia-small.tar
  measurement          mean ± σ            min … max           outliers         delta
  wall_time          119ms ± 1.97ms     117ms …  128ms          1 ( 2%)        0%
  peak_rss          26.7MB ± 85.7KB    26.6MB … 26.9MB          0 ( 0%)        0%
  cpu_cycles         406M  ± 4.67M      399M  …  424M           1 ( 2%)        0%
  instructions       660M  ±  469       660M  …  660M           0 ( 0%)        0%
  cache_references  8.06M  ± 1.31M     5.65M  … 11.3M           0 ( 0%)        0%
  cache_misses       461K  ± 36.5K      433K  …  555K           5 (12%)        0%
  branch_misses     3.59M  ± 6.42K     3.58M  … 3.61M           1 ( 2%)        0%
Benchmark 2 (43 runs): removed-bounds/release/examples/compress 1 rs silesia-small.tar
  measurement          mean ± σ            min … max           outliers         delta
  wall_time          118ms ± 2.53ms     115ms …  127ms          3 ( 7%)          -  1.3% ±  0.8%
  peak_rss          26.8MB ± 77.9KB    26.7MB … 27.0MB          0 ( 0%)          +  0.2% ±  0.1%
  cpu_cycles         400M  ± 8.00M      391M  …  437M           2 ( 5%)          -  1.4% ±  0.7%
  instructions       623M  ±  522       623M  …  623M           0 ( 0%)        ⚡ -  5.6% ±  0.0%
  cache_references  7.91M  ± 1.45M     5.89M  … 11.9M           1 ( 2%)          -  1.9% ±  7.4%
  cache_misses       458K  ± 29.1K      433K  …  550K           1 ( 2%)          -  0.5% ±  3.1%
  branch_misses     3.34M  ± 7.35K     3.33M  … 3.36M           0 ( 0%)        ⚡ -  6.8% ±  0.1%
```

https://blog.readyset.io/bounds-checks/

https://github.com/andrewrk/poop

Lossy Compression

622kb

Atlantic Ghost Crab © Hans Hillewaert

Lossy Compression

622kb ➡ 87kb