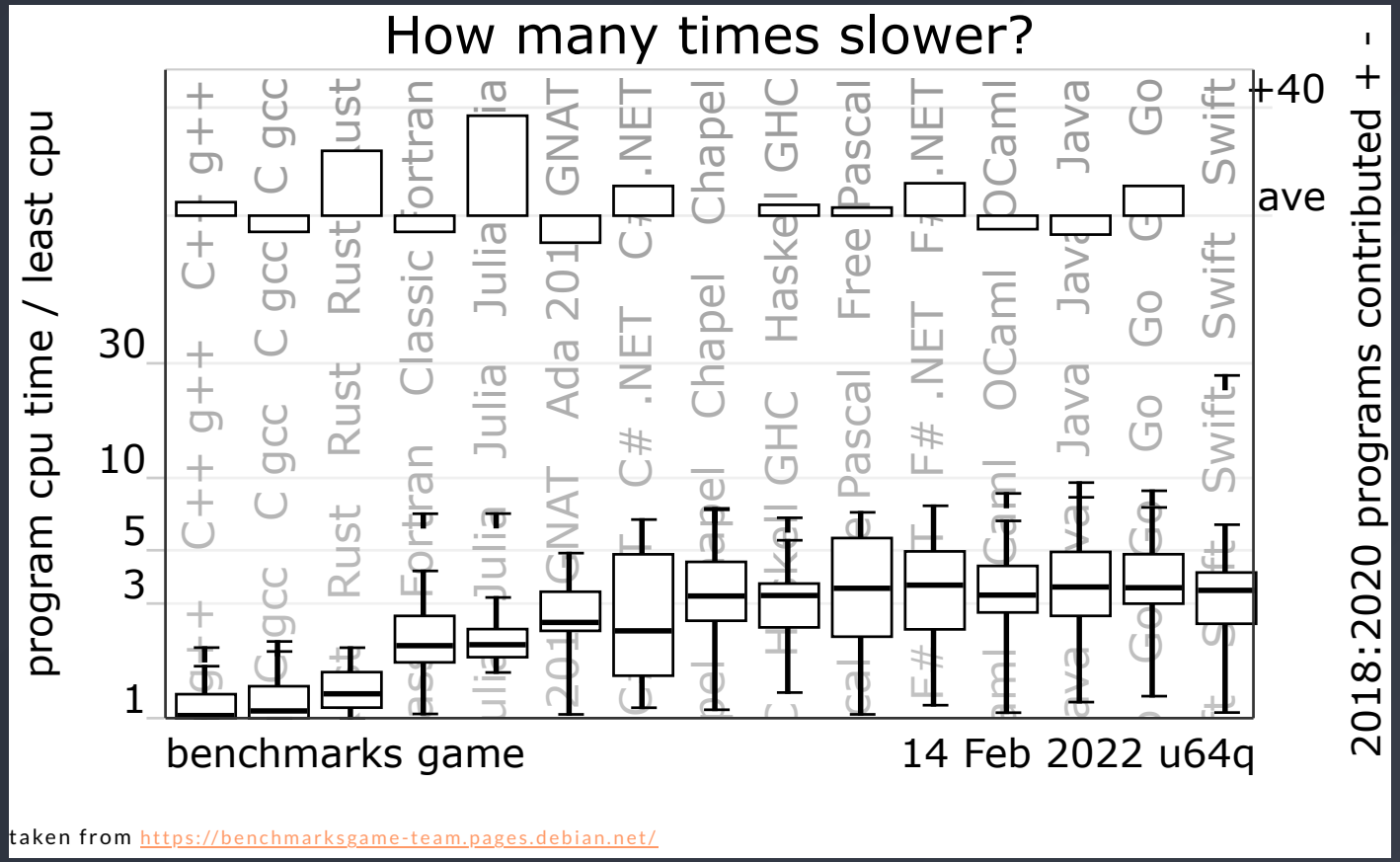
Rust

What, why

MDAP Hacky Hour

What is Rust

- Systems programming language (like C, C++, Fortran, etc.)
- No runtime or garbage collection (like Python, Julia, Java etc.)
- Statically typed (i32, f64, u8, etc.)
- **Memory and thread safe**
- Fast



taken from https://benchmarksgame-team.pages.debian.net/

Safety

- ***Memory safety***
  - Approximately 70% if the security issues dealt with by Microsoft are memory safety related.
    - e.g. overflow, dangling pointers, races
  - Important for critical systems (and research software!)
- ***"Fearless concurrency"***
  - Almost guaranteed free from data races with safe code

A programmer had a problem. They thought to themselves,
"I know, I'll solve it with threads!".

have Now problems. two they

-- anon

Safety

The borrow checker

- In Rust, ownership is tracked and checked at compile time

```
let mut s = String::from("hello");

let r1 = &s; // no problem
let r2 = &s; // no problem
println!("{} and {}", r1, r2);
// variables r1 and r2 will not be used after this point

let r3 = &mut s; // no problem
println!("{}", r3);
```

example taken from "the book"

Safety

The borrow checker

```
$ cargo run
   Compiling ownership v0.1.0 (file:///projects/ownership)
error[E0502]: cannot borrow `s` as mutable because it is also borrowed as immutable
 --> src/main.rs:6:14
  |
4 |     let r1 = &s; // no problem
  |              -- immutable borrow occurs here
5 |     let r2 = &s; // no problem
6 |     let r3 = &mut s; // BIG PROBLEM
  |              ^^^^^^ mutable borrow occurs here
7 |
8 |     println!("{}, {}, and {}", r1, r2, r3);
  |                                -- immutable borrow later used here

For more information about this error, try `rustc --explain E0502`.
error: could not compile `ownership` due to previous error
```

Why not Rust

- It's a complicated and large language (though still *way* smaller than C++)
- It takes time to learn and the early phases can be painful
- The eco-system is relatively immature compared to older similar langauges
  - Missing or incomplete scientific libraries.
  - Are we *X* yet?

Why Rust

- Enforces good practices and safety
  - Important for research accuracy and reproducibility
  - Will improve your code quality in all languages (especially other systems-level)

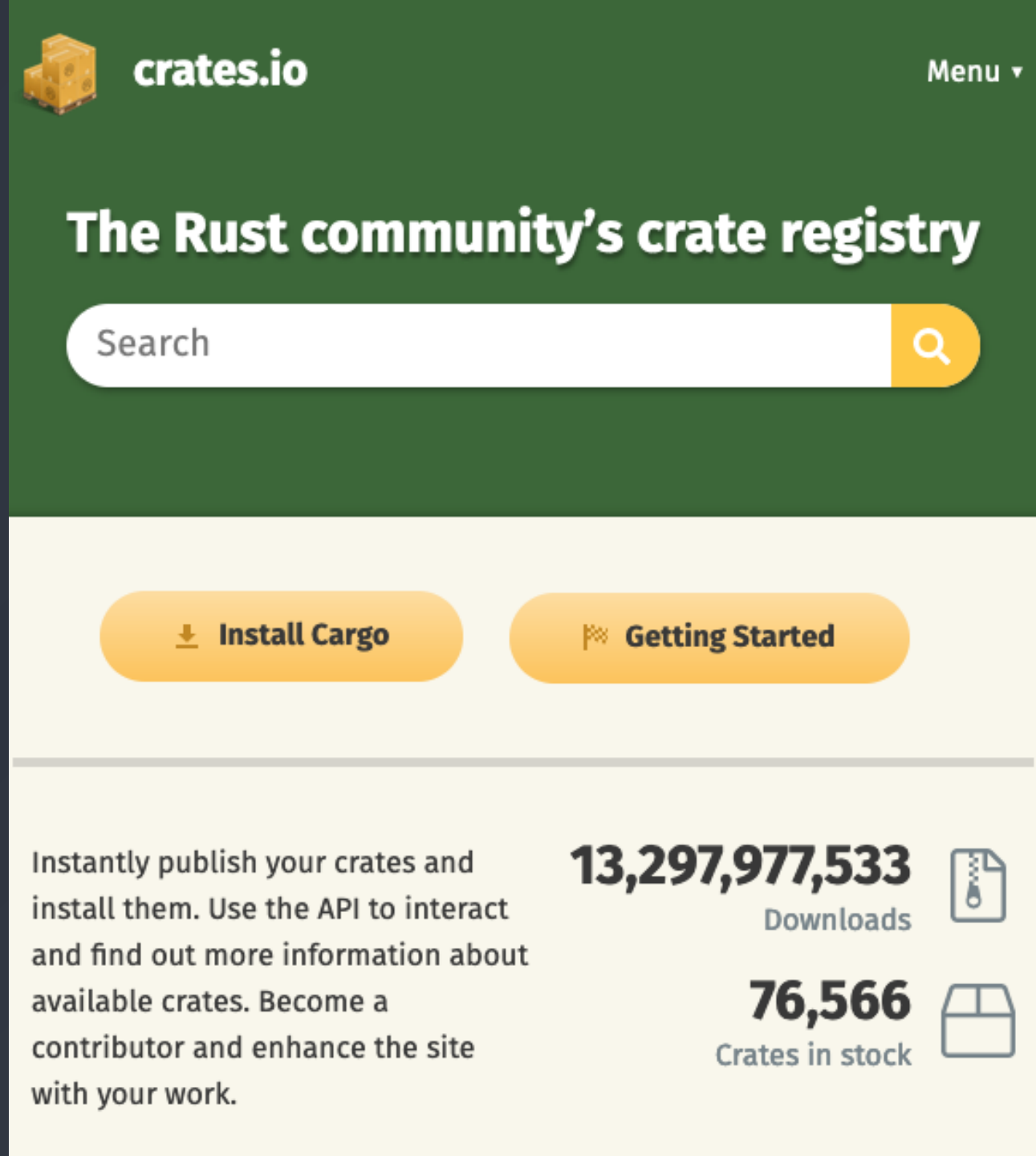
- **amazing tooling** (package manager, compiler, linter, language-server)
- Excellent learning resources ("the book", rustlings etc.)
- Good standard library (everything you'd expect to see)
- Loved by many! (e.g. most loved language in StackOverflow Developer Survey every year since 2016)
- Passionate and welcoming community

Modern tooling

Cargo

```toml
[package]
name = "rtiow"
version = "0.1.0"
edition = "2021"

[dependencies]
image = "0.23"
nalgebra-glm = "0.15"
palette = { git = "https://github.com/Ogeon/palette.git" }
indicatif = "0.16"
rand = "0.8"
rayon = "1.5.1"
```



Modern tooling

Rust analyzer

- Type hints
- Refactoring
- Diagnostics
- Documentation

```rust
fn hit(&self, ray: &Ray, t_min: f32, t_max: f32) -> Option<HitRecord> {
    match *self {
        Hittable::Sphere {
            centre,
            radius,
            ref material,
        } => {
            let oc = ray.origin - centre;
            let a = ray.norm_squared();
            let half_b = oc.dot(&ray.direction);
            let c = oc.norm_squared() - radius * &radius;  f32     taken reference of right operand `#[warn(clippy::op_ref)]` on by default for
            let discriminant = half_b * half_b - a  Diagnostics:
                                                    1. taken reference of right operand
                                                       `#[warn(clippy::op_ref)]` on by default
                                                       for further information visit https://rust-lang.github.io/rust-clippy/master/index.html#op_ref
            if discriminant < 0.0 {                 2. use the right value directly: `radius`
                return None;
            }

            // Find the nearest root that lies in an acceptible range (t_min < t < t_max)
            let sqrtd = discriminant.sqrt();
            let mut root = (-half_b - sqrtd) / a;
            if root < t_min || t_max < root {
                root = (-half_b + sqrtd) / a;
                if root < t_min || t_max < root {
                    return None;
                }
            }

            let t = root;
            let pos = ray.at(t);
            let outward_normal = (pos - centre) / radius;
            Some(HitRecord::new(ray, pos, outward_normal, t, material))
        }
    }
}
```

Change in development cycle

C, C++, Fortran, etc.

- Compile → fix errors → compile → fix errors → [...]
- run → seg fault → spend days trying to find overflow bug → crack out gdb + valgrind
- test → fix silent bugs
- test → fix logic bugs
- production!

Change in development cycle

Rust

- Fix errors and warnings as you write your code
- test -> fix logic bugs
- production!

MDAP Hacky Hour