PV281: Programování v Rustu

ORGANIZAČNÍ INFORMACE

PŘEDNÁŠKY

2h týdně

Přednášející: **Grolig, Pitner**

Během přenášky projdeme teorii a základní příklady.

Jsou nahrávané a dostupné na:
https://www.dropbox.com/sh/cvlth71b81x4raf/AABmXNUWMq-QSjNz4sbDOs2Ea?dl=0

Budou streamované v **úterý večer (20:00)**.
https://www.twitch.tv/vexfalard

CVIČENÍ

2h týdně, 12 cvičení

Cvičící: **Pitoňáková**

Na cvičení si zopakujete probranou látku a budete vypracovávat příklady s pomocí cvičí.

Organizační informace, diskuze k Rustu, pomoc s projekty, hledání parťáka

https://discord.gg/ZMvRqSqTfZ

HODNOCENÍ

Maximum bodů: **100**
Minimum pro úspěšné zakončení předmětu: **75**

HODNOCENÍ: Cvičení

- 10 bodů aktivita na cvičení
- + bonusové body
- neřešíme absenci, ale v případě absence není možné získat body

HODNOCENÍ: Úkoly

- 26 bodů domácí úkoly
- 12 bodů za code review

HODNOCENÍ: Úkoly

První 4 úkoly za 2 body

- Úkoly obsahují automatizované testy. Procházející testy = 2b.
- Hodnocení kódu od opravujících (Katka & Dan) = 2b po zapracování připomínek

HODNOCENÍ: Úkoly

Zbývajících 6 úkolů za 3 body, ke každému code review za 2 body

- Úkoly opět obsahují automatizované testy. Procházející testy = 2b.
- Na cvičení si najdete reviewera z řad spolužáků (ze stejného cvičení)
- Reviewer projde Váš kód a zpracuje code review (CR) v pull requestu.

HODNOCENÍ: Úkoly

Zbývajících 6 úkolů za 3 body, ke každému code review za 2 body

- Zapracujete změny (0,5b)
- Reviewer zkontroluje změny a pokud to bude nutné, tak dá druhé kolo připomínek.
- Zapracujete změny pokud budou nutné (0,5b)
- Opravující zhodnotí kvalitu provedeného CR a reviewer může dostat 2 body

HODNOCENÍ: Code Review

Každé code review je za 2 body

- Opravující hodnotí kvalitu code review z pohledu:
  - odhalených nedostatků
  - úroveň komunikace
  - kvalita vysvětlení

HODNOCENÍ: Projekt

- **52 bodů** za týmový projekt zakončený obhajobou
- **3-4 členné týmy** - nikdy ne více. V případě menšího týmu nedojde ke snížení náročnosti.
- Týmy lze tvořit mezi skupinami
- Zadání zveřejníme v **5. týdnu** semestru

BONUSOVÉ BODY

- Možné získat na cvičeních za aktivity
- Při zpracování úkolů/projektu za špičkové zpracování a výrazné překročení rozsahu
- Na přednášce za aktivitu
- Seznámení ostatních na přednášce s vlastním zajímavým projektem

PROBÍRANÁ LÁTKA

- Intro - trocha historie, hlavní výhody a nevýhody
- Syntaxe jazyka
- CLI aplikace
- Asynchronní programování
- Paralelní programování
- Práce se soubory
- Obsluha chybových stavů

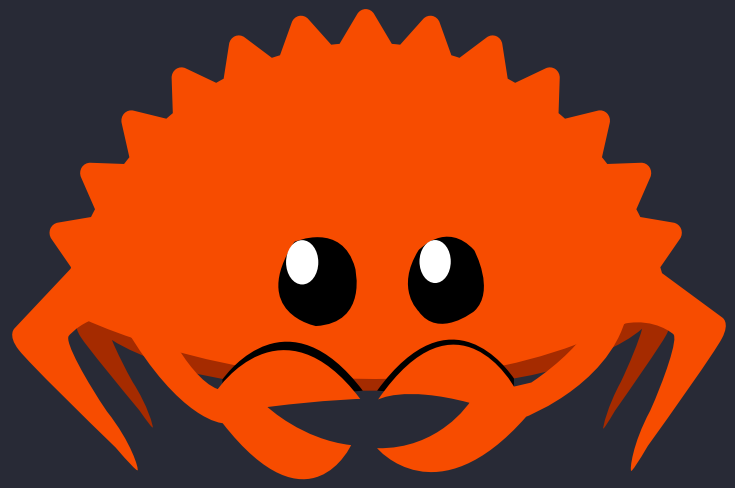
PROBÍRANÁ LÁTKA

- Práce s databází
- Webové frameworky
- GRPC protokol
- Spolupráce s jinými jazyky (C, Python, Javascript)
- Desktopové aplikace
- Webassembly

PROBÍRANÁ LÁTKA MIMO RUST

- Budeme hodně pracovat s GITem a Gitlabem
- Budeme psát čisté SQL

Nyní už o Rustu

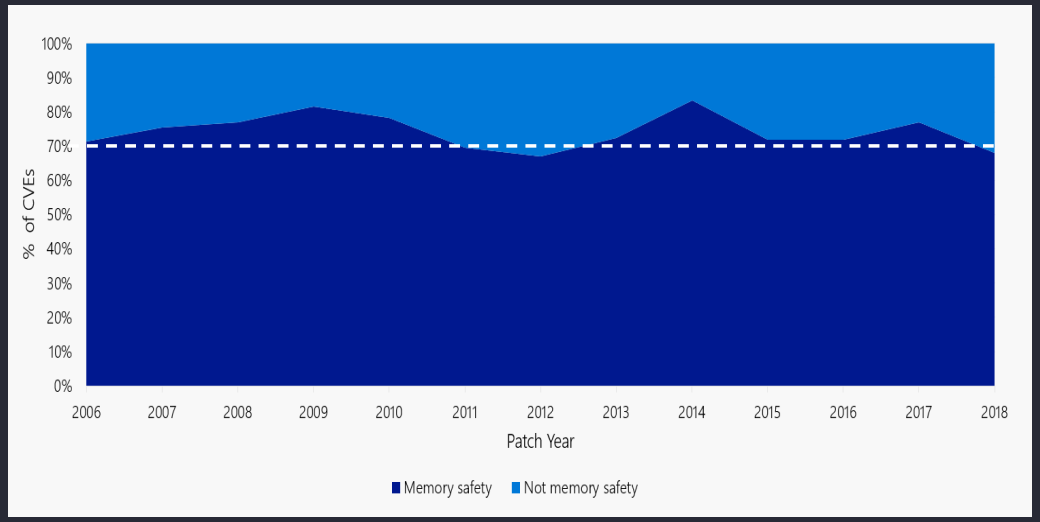


Seznamte se:

Krab Ferris

Historie

- Rust vznikl v Mozille
- Cílem bylo vytvořit jazyk bez garbage kolekce s bezpečnou prací s pamětí



70% bezpečnostních děr v Microsoftu bylo spojených s prací s pamětí.

https://msrc-blog.microsoft.com/2019/07/16/a-proactive-approach-to-more-secure-code/

Rust Foundation

Mozilla ale dál nezvládala sama rozvíjet Rust. To vedle ke vzniku Rust Foundation v roce 2020.

Zakládajícími členy se stali:

Další známé firmy využívající Rust

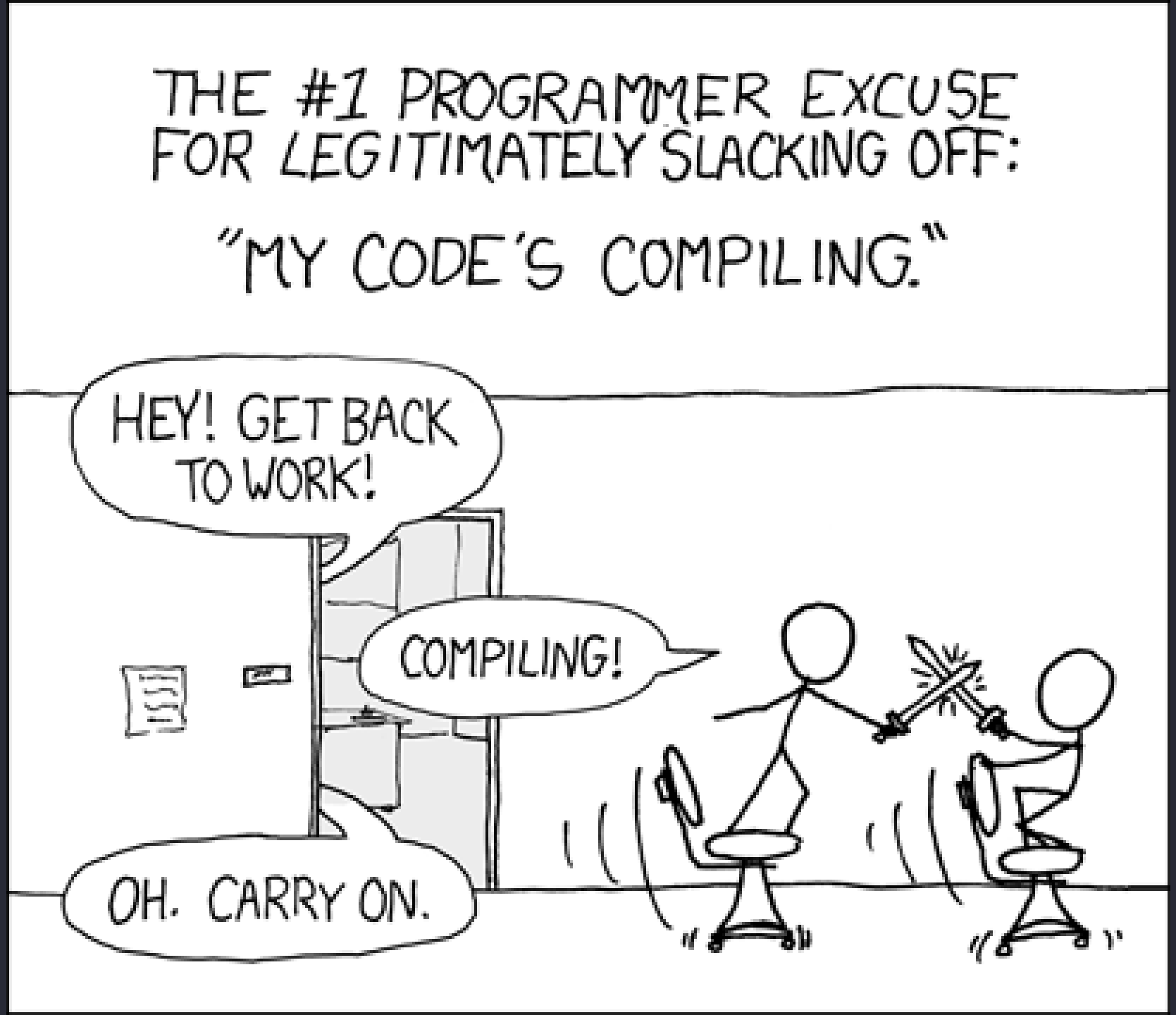
Tehle seznam se už hodně natáhl, ale můžeme jmenovat:
1Password, Apple, Canonical, Cloudflare, Discord, Dropbox, Figma, Facebook, System76, OVH

Hlavní výhodu Rustu

Bezpečnost

V jazycích typu C vznikají problémy s manuální správou pamětí jako dangling pointer, dvojité uvolnění aj. V moderním C++ je spousta věcí řešena technikami jako je RAII nebo smartpointy, ale v Rustu to nehlídá člověk, ale překladač.

Cenou za to je délka překladu.


THE #1 PROGRAMMER EXCUSE FOR LEGITIMATELY SLACKING OFF:
"MY CODE'S COMPILING."
HEY! GET BACK TO WORK!
COMPILING!
OH. CARRY ON.

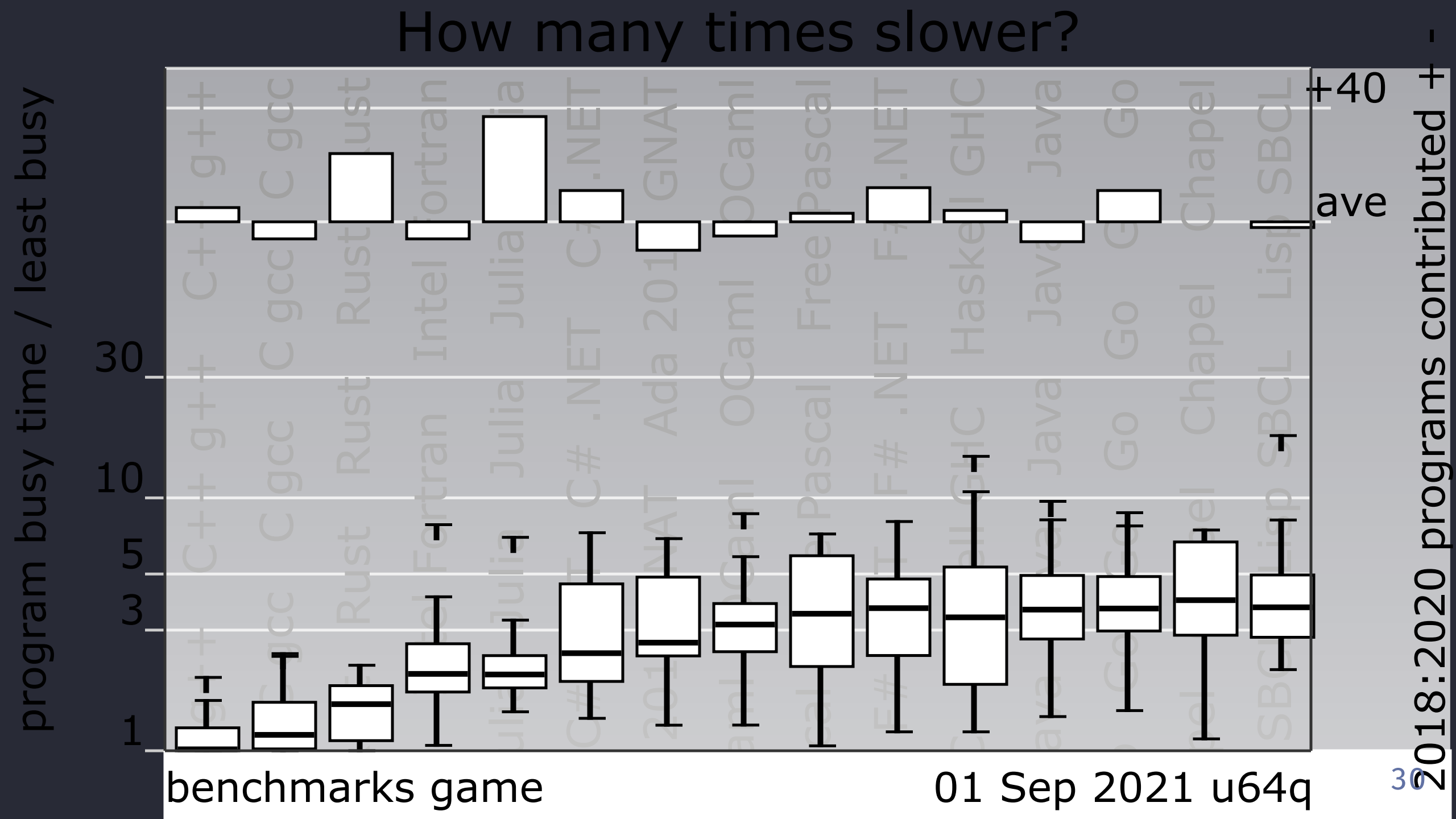
Rychlost

Prakticky všechny jazyky jsou pomalejší než C a C++. Je to kvůli abstrakcím, garbage kolekci aj.

To řeší:

- zero cost abstrakce
- správa paměti během kompilace

How many times slower?



Pohled oproti C

reverse-complement

source	secs	mem	gz	busy	cpu load
Rust	**0.45**	498,964	3040	0.77	25% 23% 100% 25%
C gcc	0.86	712,208	820	1.27	99% 28% 1% 19%

binary-trees

source	secs	mem	gz	busy	cpu load
Rust	**1.09**	198,728	765	3.90	87% 98% 88% 86%
C gcc	1.54	168,832	809	4.35	60% 67% 57% 100%

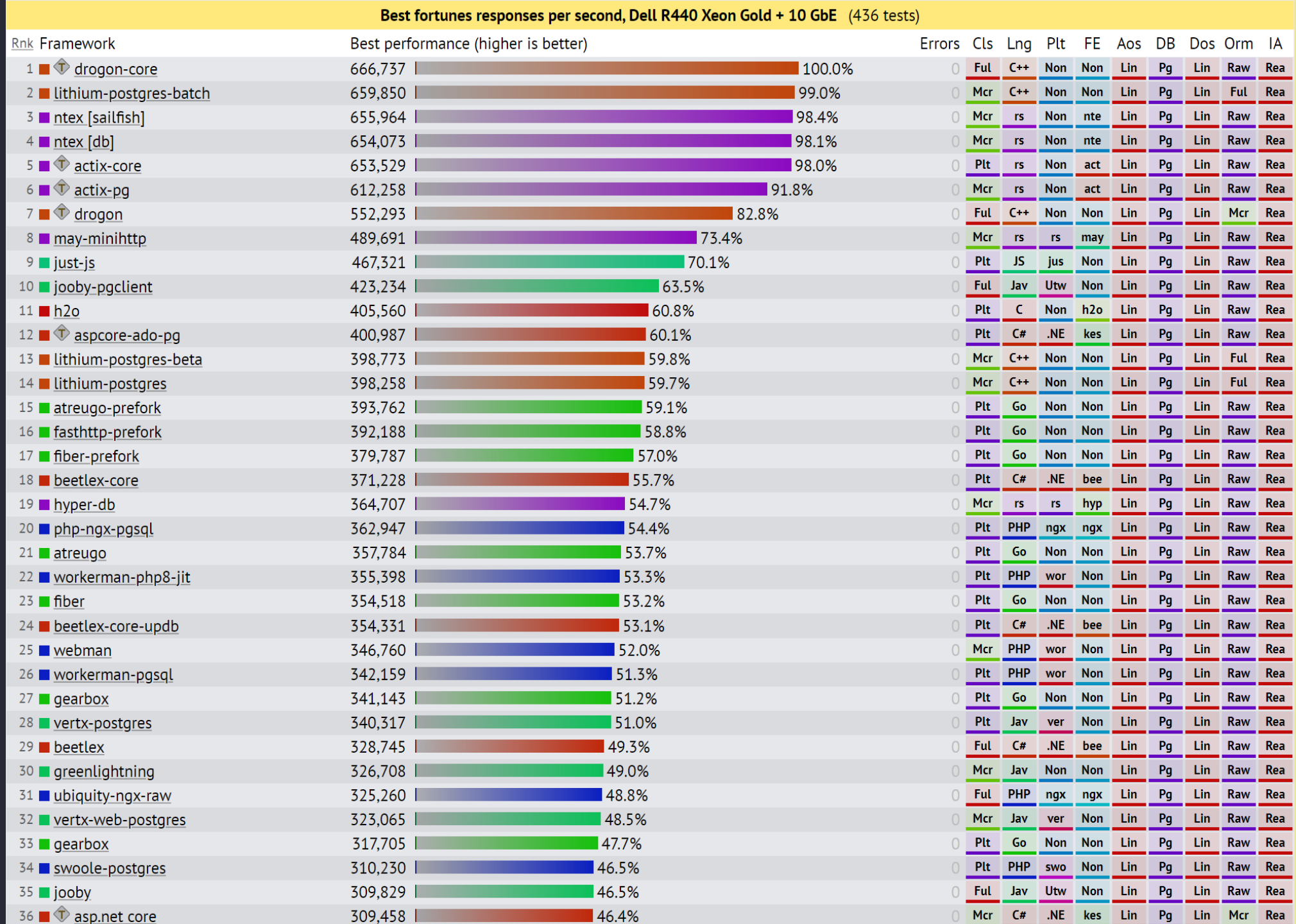
mandelbrot

source	secs	mem	gz	busy	cpu load
Rust	**0.93**	32,676	763	3.70	100% 99% 100% 100%
C gcc	1.27	31,792	1135	5.08	100% 100% 99% 100%

regex-redux

source	secs	mem	gz	busy	cpu load
Rust	**0.77**	147,524	2458	1.99	54% 59% 91% 54%
C gcc	0.80	152,172	1397	2.01	52% 99% 48% 53%

Porovnání frameworků ve Fortunes


Best fortunes responses per second, Dell R440 Xeon Gold + 10 GbE (436 tests)
Rnk Framework Best performance (higher is better) Errors Cls Lng Plt FE Aos DB Dos Orm IA
1 drogon-core 666,737 100.0% 0 Ful C++ Non Non Lin Pg Lin Raw Rea
2 lithium-postgres-batch 659,850 99.0% 0 Mcr C++ Non Non Lin Pg Lin Ful Rea
3 ntex [sailfish] 655,964 98.4% 0 Mcr rs Non nte Lin Pg Lin Raw Rea
4 ntex [db] 654,073 98.1% 0 Mcr rs Non nte Lin Pg Lin Raw Rea
5 actix-core 653,529 98.0% 0 Plt rs Non act Lin Pg Lin Raw Rea
6 actix-pg 612,258 91.8% 0 Mcr rs Non act Lin Pg Lin Raw Rea
7 drogon 552,293 82.8% 0 Ful C++ Non Non Lin Pg Lin Mcr Rea
8 may-minihttp 489,691 73.4% 0 Mcr rs rs may Lin Pg Lin Raw Rea
9 just-js 467,321 70.1% 0 Plt JS jus Non Lin Pg Lin Raw Rea
10 jooby-pgclient 423,234 63.5% 0 Ful Jav Utw Non Lin Pg Lin Raw Rea
11 h2o 405,560 60.8% 0 Plt C Non h2o Lin Pg Lin Raw Rea
12 aspcore-ado-pg 400,987 60.1% 0 Plt C# .NE kes Lin Pg Lin Raw Rea
13 lithium-postgres-beta 398,773 59.8% 0 Mcr C++ Non Non Lin Pg Lin Ful Rea
14 lithium-postgres 398,258 59.7% 0 Mcr C++ Non Non Lin Pg Lin Ful Rea
15 atreugo-prefork 393,762 59.1% 0 Plt Go Non Non Lin Pg Lin Raw Rea
16 fasthttp-prefork 392,188 58.8% 0 Plt Go Non Non Lin Pg Lin Raw Rea
17 fiber-prefork 379,787 57.0% 0 Plt Go Non Non Lin Pg Lin Raw Rea
18 beetlex-core 371,228 55.7% 0 Plt C# .NE bee Lin Pg Lin Raw Rea
19 hyper-db 364,707 54.7% 0 Mcr rs rs hyp Lin Pg Lin Raw Rea
20 php-ngx-pgsql 362,947 54.4% 0 Plt PHP ngx ngx Lin Pg Lin Raw Rea
21 atreugo 357,784 53.7% 0 Plt Go Non Non Lin Pg Lin Raw Rea
22 workerman-php8-jit 355,398 53.3% 0 Plt PHP wor Non Lin Pg Lin Raw Rea
23 fiber 354,518 53.2% 0 Plt Go Non Non Lin Pg Lin Raw Rea
24 beetlex-core-updb 354,331 53.1% 0 Plt C# .NE bee Lin Pg Lin Raw Rea
25 webman 346,760 52.0% 0 Mcr PHP wor Non Lin Pg Lin Raw Rea
26 workerman-pgsql 342,159 51.3% 0 Plt PHP wor Non Lin Pg Lin Raw Rea
27 gearbox 341,143 51.2% 0 Plt Go Non Non Lin Pg Lin Raw Rea
28 vertx-postgres 340,317 51.0% 0 Plt Jav ver Non Lin Pg Lin Raw Rea
29 beetlex 328,745 49.3% 0 Ful C# .NE bee Lin Pg Lin Raw Rea
30 greenlightning 326,708 49.0% 0 Mcr Jav Non Non Lin Pg Lin Raw Rea
31 ubiquity-ngx-raw 325,260 48.8% 0 Ful PHP ngx ngx Lin Pg Lin Raw Rea
32 vertx-web-postgres 323,065 48.5% 0 Mcr Jav ver Non Lin Pg Lin Raw Rea
33 gearbox 317,705 47.7% 0 Plt Go Non Non Lin Pg Lin Raw Rea
34 swoole-postgres 310,230 46.5% 0 Plt PHP swo Non Lin Pg Lin Raw Rea
35 jooby 309,829 46.5% 0 Ful Jav Utw Non Lin Pg Lin Raw Rea
36 asp.net core 309,458 46.4% 0 Mcr C# .NE kes Lin Pg Lin Mcr Rea

299	wildfly-ee	28,517	4.3%	0	Ful	Jav	Svt	Wil	Lin	My	Lin	Ful	Rea
300	sinatra-sequel-postgres-passenger-mri	28,467	4.3%	0	Mcr	Rby	Rac	Pas	Lin	Pg	Lin	Ful	Rea
301	sinatra-sequel-passenger-mri	27,995	4.2%	0	Mcr	Rby	Rac	Pas	Lin	My	Lin	Ful	Rea
302	elixir-plug-ecto	27,632	4.1%	0	Mcr	Eli	bea	cow	Lin	Pg	Lin	Ful	Rea
303	roda-sequel-unicorn-mri	27,090	4.1%	0	Mcr	Rby	Rac	Uni	Lin	My	Lin	Ful	Rea
304	aqueduct	27,080	4.1%	0	Mcr	Dar	Non	Non	Lin	Pg	Lin	Mcr	Rea
305	api_hour	26,683	4.0%	0	Mcr	Py	asy	Gun	Lin	Pg	Lin	Raw	Rea
306	yii2-raw	26,627	4.0%	0	Ful	PHP	fpm	ngx	Lin	My	Lin	Raw	Rea
307	quart-uvicorn	26,446	4.0%	0	Mcr	Py	Non	uvi	Lin	Pg	Lin	Raw	Rea
308	sinatra-sequel-postgres-torquebox-jruby	24,933	3.7%	0	Mcr	Rby	Rac	Tor	Lin	Pg	Lin	Ful	Rea
309	cppcms-postgres	24,787	3.7%	0	Plt	C++	Non	Non	Lin	Pg	Lin	Raw	Rea
310	express-graphql-mysql	24,535	3.7%	0	Mcr	JS	njs	Non	Lin	My	Lin	Ful	Rea
311	duct-httpkit	24,327	3.6%	0	Mcr	Clj	Rin	Non	Lin	Pg	Lin	Raw	Rea
312	cppcms	23,979	3.6%	0	Plt	C++	Non	Non	Lin	My	Lin	Raw	Rea
313	codeigniter	23,911	3.6%	0	Ful	PHP	fpm	ngx	Lin	My	Lin	Raw	Rea
314	roda-sequel-postgres-unicorn-mri	23,709	3.6%	0	Mcr	Rby	Rac	Uni	Lin	Pg	Lin	Ful	Rea
315	flask-raw	23,573	3.5%	0	Mcr	Py	Mei	Non	Lin	My	Lin	Raw	Rea
316	fat-free-raw	23,534	3.5%	0	Ful	PHP	fpm	ngx	Lin	My	Lin	Raw	Rea
317	spring	23,401	3.5%	0	Ful	Jav	tom	Non	Lin	Pg	Lin	Mcr	Rea
318	hamlet	23,125	3.5%	0	Ful	PHP	fpm	ngx	Lin	My	Lin	Mcr	Rea
319	sinatra-sequel-postgres-unicorn-mri	22,580	3.4%	0	Mcr	Rby	Rac	Uni	Lin	Pg	Lin	Ful	Rea
320	racket	22,568	3.4%	0	Mcr	rac	rac	rac	Lin	Pg	Lin	Raw	Rea
321	aiohttp	22,561	3.4%	0	Mcr	Py	asy	Gun	Lin	Pg	Lin	Ful	Rea
322	spring-mongo	22,491	3.4%	0	Ful	Jav	tom	Non	Lin	Mo	Lin	Ful	Rea
323	sinatra-sequel-torquebox-jruby	22,418	3.4%	0	Mcr	Rby	Rac	Tor	Lin	My	Lin	Ful	Rea
324	spring-webflux-pgclient	22,234	3.3%	0	Ful	Jav	Nty	Non	Lin	Pg	Lin	Mcr	Rea
325	lumen-laravel-s	22,210	3.3%	0	Mcr	PHP	swo	Non	Lin	My	Lin	Ful	Rea
326	phalcon	21,851	3.3%	0	Ful	PHP	fpm	ngx	Lin	My	Lin	Raw	Rea
327	spring-webflux-jdbc	21,830	3.3%	0	Ful	Jav	Nty	Non	Lin	Pg	Lin	Mcr	Rea
328	spring-jpa	21,685	3.3%	0	Ful	Jav	tom	Non	Lin	Pg	Lin	Ful	Rea
329	sinatra-sequel-unicorn-mri	21,648	3.2%	0	Mcr	Rby	Rac	Uni	Lin	My	Lin	Ful	Rea
330	pyramid-py2	21,493	3.2%	0	Ful	Py	Non	Mei	Lin	Pg	Lin	Ful	Rea
331	ninja-standalone	21,024	3.2%	0	Ful	Jav	Jty	Non	Lin	My	Lin	Ful	Rea
332	spring-webflux-mongo	20,755	3.1%	0	Ful	Jav	Nty	Non	Lin	Mo	Lin	Ful	Rea
333	tornado-py3-uvloop	20,538	3.1%	0	Plt	Py	Non	Tor	Lin	Pg	Lin	Raw	Rea
334	api_hour-mysql	19,846	3.0%	0	Mcr	Py	asy	Gun	Lin	My	Lin	Raw	Rea
335	pyramid	19,608	2.9%	0	Ful	Py	Non	Mei	Lin	Pg	Lin	Ful	Rea
336	lumen-swoole	18,860	2.8%	0	Mcr	PHP	swo	Non	Lin	My	Lin	Ful	Rea
337	php-eloquent	18,462	2.8%	0	Plt	PHP	fpm	ngx	Lin	My	Lin	Ful	Rea
338	dropwizard-mongodb	18,369	2.8%	0	Ful	Jav	JAX	Jty	Lin	Mo	Lin	Ful	Rea
339	fat-free	18,197	2.7%	0	Ful	PHP	fpm	ngx	Lin	My	Lin	Ful	Rea
340	flask-pypy2-raw	18,121	2.7%	0	Mcr	Py	Tor	Non	Lin	My	Lin	Raw	Rea
341	hapi-postgres	17,794	2.7%	0	Mcr	JS	njs	Non	Lin	Pg	Lin	Ful	Rea

Konkurence

Při paralelním programování často dochází k všemožným problémům. Při překladu, díky síle překladače (rozumějte statickým kontrolám), poskytuje podstatně větší jistotu.

Další výhody

- jazyk nemá dědičnost, místo toho vyžaduje kompozici
- moderní a pokrokový tooling
  - balíčkovací manager
  - neexistující null
  - dokumentování kódu
  - testování

POVÍDLA O SILNÉM TYPOVÁNÍ A STATICKY TYPOVANÉM JAZYCE

Nevýhody

- Pomalý překlad
- Stále ještě není tolik knihoven, abysme rychle zvládli všechno.
- Často více psaní než v jiných jazycích

Jak se rozhodovat při výběru jazyka?

Potřebuju to jen rychle **naprasit -> Python**
Bude to běžet v prohlížeči a nebo je to **UI -> Javascript/Typescript**
Můžu použít **managovaný jazyk -> Golang** > C#, Kotlin > Java
Jinak vyberu Rust

Příběh Discordu

As usual with a garbage collected language the problem was CPU stalls due to garbage collection spikes. But it non-GC languages you have to worry about memory fragmentation, especially for long lived processes. When you get that sev 1 bug that happens after two months of flawless execution it will often be a memory allocation failure due to memory fragmentation. So you end up creating your own memory allocator anyway.

Příběh Discordu

When we started load testing, we were instantly pleased with the results. The latency of the Rust version was just as good as Go's and had no latency spikes! Remarkably, we had only put very basic thought into optimization as the Rust version was written. Even with just basic optimization, Rust was able to outperform the hyper hand-tuned Go version.

Příběh Discordu

After a bit of profiling and performance optimizations, we were able to beat Go on every single performance metric. Latency, CPU, and memory were all better in the Rust version.

Příběh Discordu

Along with performance, Rust has many advantages for an engineering team. For example, its type safety and borrow checker make it very easy to refactor code as product requirements change or new learnings about the language are discovered. Also, the ecosystem and tooling are excellent and have a significant amount of momentum behind them.

Also, Our business case for using Go - it's all about saving money.

http://highscalability.com/blog/2020/2/7/stuff-the-internet-says-on-scalability-for-february-7th-2020.html

https://blog.discord.com/why-discord-is-switching-from-go-to-rust-a190bbca2b1f

Verzování Rustu

nightly: vydáváno každodenně

beta: vydávána jednou za 6 týdnů

stable: vydávána jednou za 6 týdnů (následujících po betě)

Problémy verzování

Dříve nebyly všechny features dostupné ve stabilní verzi. Tvůrci frameworků proto sahali po nightly a ta musela být používaná i v projektu.

Dnes už to neplatí a použití nightly verze se snažíme vyhnout.

Instalace Rustu

Na Unixu

```
curl --proto '=https' --tlsv1.2 -sSf https://sh.rustup.rs | sh
```

Na Windows

stáhnout rustup-init.exe

https://static.rust-lang.org/rustup/dist/x86_64-pc-windows-msvc/rustup-init.exe

Základní nástroje

rustup spravuje verze Rustu

rustc překladač

cargo balíčkovací systém spravující projekt a závislosti

Založení nového projektu

```
cargo new nazev_projektu
```

tím se vytvoří

```
nazev_projektu
+-- Cargo.toml
+-- src
|   +--main.rs
```

Překlad a spuštění

```
cargo build
cargo run
```

Cargo.toml

```
[package]
name = "hello_world"
version = "0.1.0"
authors = ["Your Name <you@example.com>"]

[dependencies]
rand = { git = "https://github.com/rust-lang-nursery/rand.git" }
```

Cargo.lock

```
[[package]]
name = "hello_world"
version = "0.1.0"
dependencies = [
 "rand 0.1.0 (git+https://github.com/rust-lang-nursery/rand.git#9f35b8e439eeedd60b9414c58f389bdc6a3284f9)",
]

[[package]]
name = "rand"
version = "0.1.0"
source = "git+https://github.com/rust-lang-nursery/rand.git#9f35b8e439eeedd60b9414c58f389bdc6a3284f9"
```

Složitější struktura

```
+-- Cargo.lock
+-- Cargo.toml
+-- src/
|   +-- lib.rs <---------------------------------- zakladní soubor knihovny
|   +-- main.rs <--------------------------------- zakladní spustitelný soubor
|   +-- bin/ <------------------------------------ veškeré další spustitelné soubory
|      +-- named-executable.rs
|      +-- another-executable.rs
|      +-- multi-file-executable/
|         +-- main.rs
|         +-- some_module.rs
+-- benches/ <---------------------------------- benchmarky
|   +-- large-input.rs
|   +-- multi-file-bench/
|      +-- main.rs
|      +-- bench_module.rs
+-- examples/ <--------------------------------- ukázky kódu
|   +-- simple.rs
|   +-- multi-file-example/
|      +-- main.rs
|      +-- ex_module.rs
+-- tests/ <------------------------------------ intergrační testy
   +-- some-integration-tests.rs
   +-- multi-file-test/
```

Ukázka kódu z main.rs

```rust
fn main() {
    println!("Hello, world!");
}
```

Datové typy

Celočíselné typy

Velikost	Znaménkový	Neznaménkový
8 bitů	i8	u8
16 bitů	i16	u16
32 bitů	i32	u32
64 bitů	i64	u64
128 bitů	i128	u128

Zápisy literálů

Velikost	Příklad
desítkové	98_222
šestnáctkové	0xff
osmičkové	0o77
binární	0b1111_0000
bajtové	b'A'

S plovoucí řádovou čárkou (IEEE-754)

Velikost	Typ
32 bitů	f32
64 bitů	f64

defaultní je f64

Boolovské typy

```
fn main() {
    let t = true;

    let f: bool = false;
}
```

Znakové typy

```
fn main() {
    let c: char = 'z';
    let z = 'ℤ';
    let heart_eyed_cat = '😻';
}
```

Složené typy

Touple (n-tice)

```
fn main() {
    let tup: (i32, f64, u8) = (500, 6.4, 1);
}
```

Pole

```
fn main() {
    let a: [i32; 5] = [1, 2, 3, 4, 5];
    let first = a[0];
    let second = a[1];

    let months = ["January", "February", "March", "April", "May", "June", "July",
            "August", "September", "October", "November", "December"];
}
```

Ovládání toku programu

Klasický if

```
fn main() {
    let number = 3;

    if number < 5 {
        println!("condition was true");
    } else if number % 3 == 0 {
        println!("number is divisible by 3");
    } else {
        println!("condition was false");
    }
}
```

Výraz s if

```
fn main() {
    let condition = true;
    let number = if condition { 5 } else { 6 };

    println!("The value of number is: {}", number);
}
```

Match

```
fn main() {
    let some_u8_value = 0u8;
    match some_u8_value {
        1 => println!("one"),
        3 => println!("three"),
        5 => println!("five"),
        7 => println!("seven"),
        _ => (),
    }
}
```

Nekonečná smyčka

```
fn main() {
    loop {
        println!("again!");
    }
}
```

While

```
fn main() {
    let mut number = 3;

    while number != 0 {
        println!("{}!", number);

        number -= 1;
    }

    println!("LIFTOFF!!!");
}
```

To je pro dnešek vše.