[TECHO] | TVP Tencent Cloud Valuable Professional intel

/serverless/DAYS

Techo × TVP 开发者峰会

/serverless/ DAYS China 2021

无服务器，大有未来

Serverless, Empower More

在腾讯云 Serverless 上部署AI推理函数

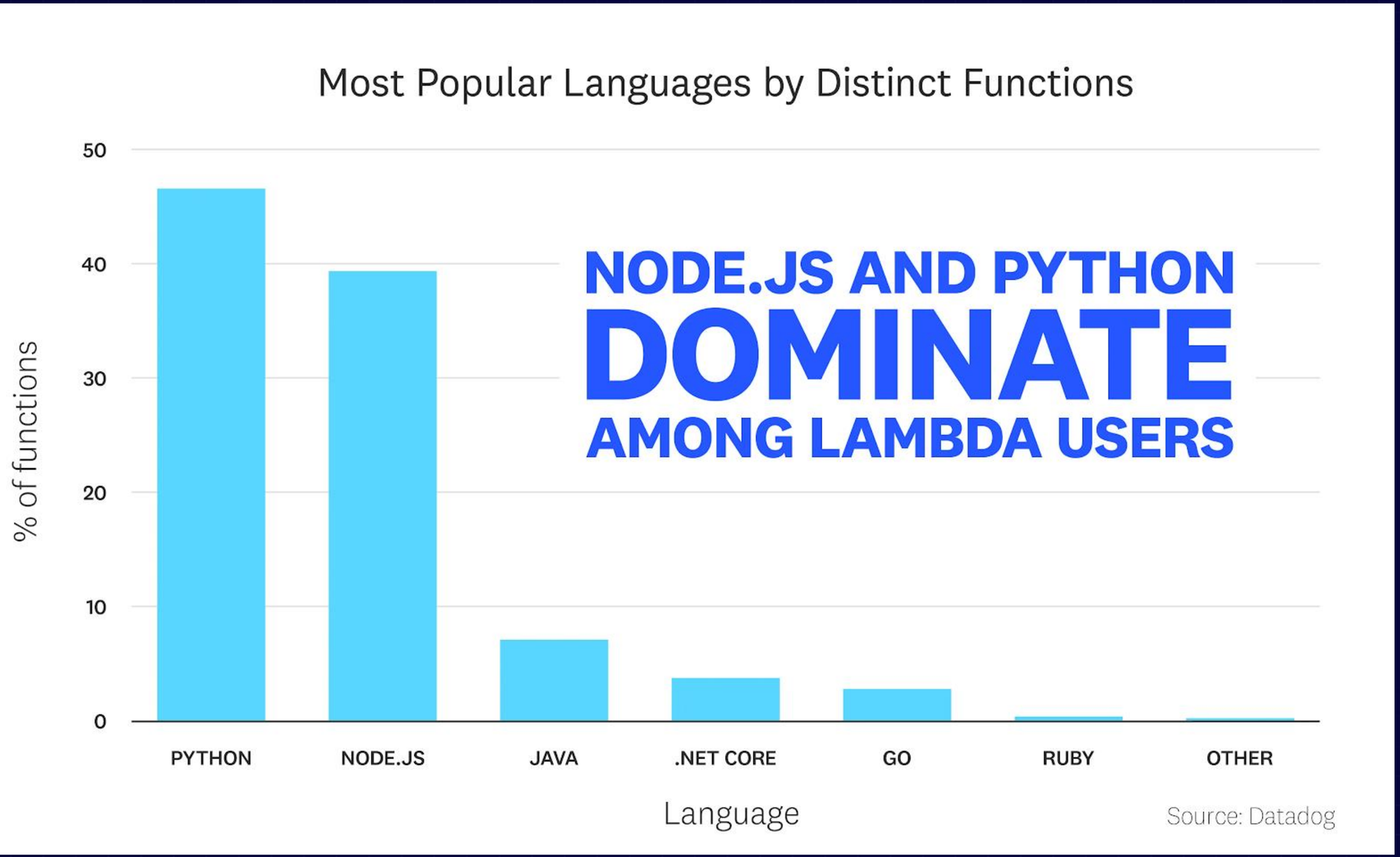
Michael Yuan, Second State

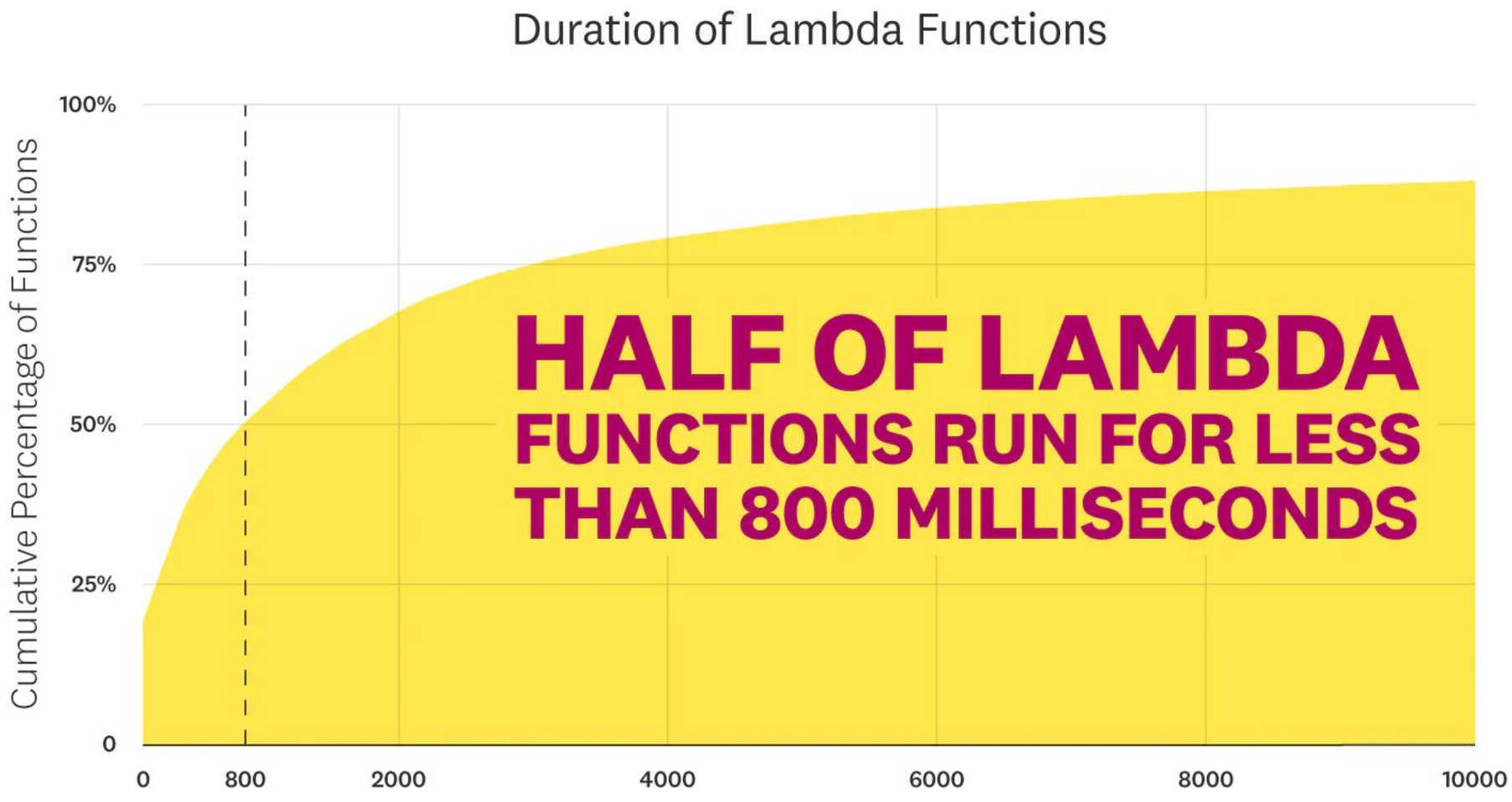
https://github.com/WasmEdge/WasmEdge

Michael Yuan

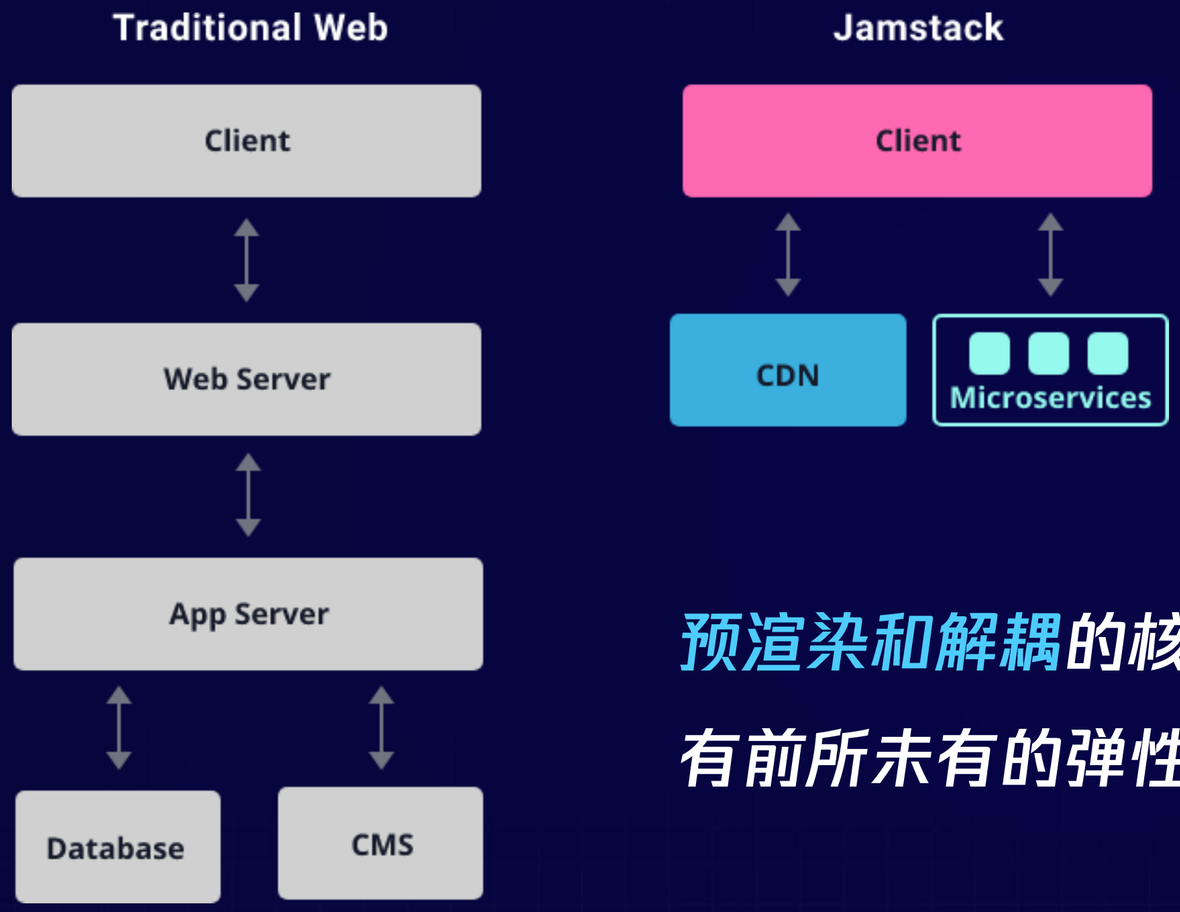
WasmEdge 维护者

Second State 创始人


Most Popular Languages by Distinct Functions
% of functions
50
40
30
20
10
0
PYTHON
NODE.JS
JAVA
.NET CORE
GO
RUBY
OTHER
Language
NODE.JS AND PYTHON
DOMINATE
AMONG LAMBDA USERS
Source: Datadog


Duration of Lambda Functions
Cumulative Percentage of Functions
100%
75%
50%
25%
0
0
800
2000
4000
6000
8000
10000
Function Duration (ms)
HALF OF LAMBDA
FUNCTIONS RUN FOR LESS
THAN 800 MILLISECONDS
Source: Datadog

最常见的 Serverless 函数用例是
在繁重的堆栈上运行一个简单的函数



预渲染和解耦的核心原则是使站点和应用程序能够有前所未有的弹性交付

Site Generators

A List of Static Site Generators for Jamstack Sites

Filter 326 Generators

All Languages | All Templates | All Licenses

Sort GitHub Stars

Next.js

★ 67892 | 12346 | ! 849

A framework for statically-exported React apps (supports server side rendering)

Hugo

★ 52019 | 5619 | ! 574

A Fast and Flexible Static Site Generator.

Gatsby

★ 50340 | 9328 | ! 286

Build blazing fast, modern apps and websites with React

为什么不用 Python

Language	Time (sec)	Memory (mb)
C++ Gcc	1.94	1.0
Rust	2.16	4.8
Java	4.03	513.8
LuaJIT	12.61	1.0
Lua 5.1	182.74	1.0
Python	314.79	4.9

COMPUTER SCIENCE

There's plenty of room at the Top: What will drive computer performance after Moore's law?

Charles E. Leiserson[1], Neil C. Thompson[1,2]*, Joel S. Emer[1,3], Bradley C. Kuszmaul[1]†, Butler W. Lampson[1,4], Daniel Sanchez[1], Tao B. Schardl[1]

Table 1. Speedups from performance engineering a program that multiplies two 4096-by-4096 matrices. Each version represents a successive refinement of the original Python code. "Running time" is the running time of the version. "GFLOPS" is the billions of 64-bit floating-point operations per second that the version executes. "Absolute speedup" is time relative to Python, and "relative speedup," which we show with an additional digit of precision, is time relative to the preceding line. "Fraction of peak" is GFLOPS relative to the computer's peak 835 GFLOPS. See Methods for more details.

Version	Implementation	Running time (s)	GFLOPS	Absolute speedup	Relative speedup	Fraction of peak (%)
1	Python	25,552.48	0.005	1	—	0.00
2	Java	2,372.68	0.058	11	10.8	0.01
3	C	542.67	0.253	47	4.4	0.03
4	Parallel loops	69.80	1.969	366	7.8	0.24
5	Parallel divide and conquer	3.80	36.180	6,727	18.4	4.33
6	plus vectorization	1.10	124.914	23,224	3.5	14.96
7	plus AVX intrinsics	0.41	337.812	62,806	2.7	40.45

为什么不是 Native

- 使用 C 写的库比较难
- 没有好的 Rust 库
- 在 Serverless 容器中比较难设置
- 不能跨不同的云提供商或云服务商移植

Rust 语言:

内存安全、高性能、高生产力

WebAssembly 运行环境:

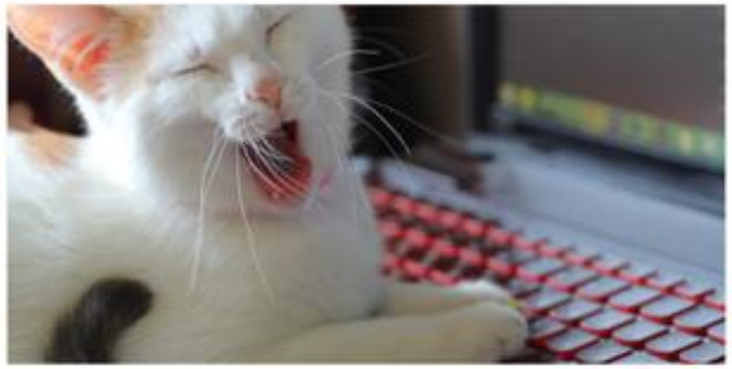
安全、可移植、高性能

{* SOFTWARE *}

Rust never sleeps: C++-alike language tops Stack Overflow survey for fourth year in a row

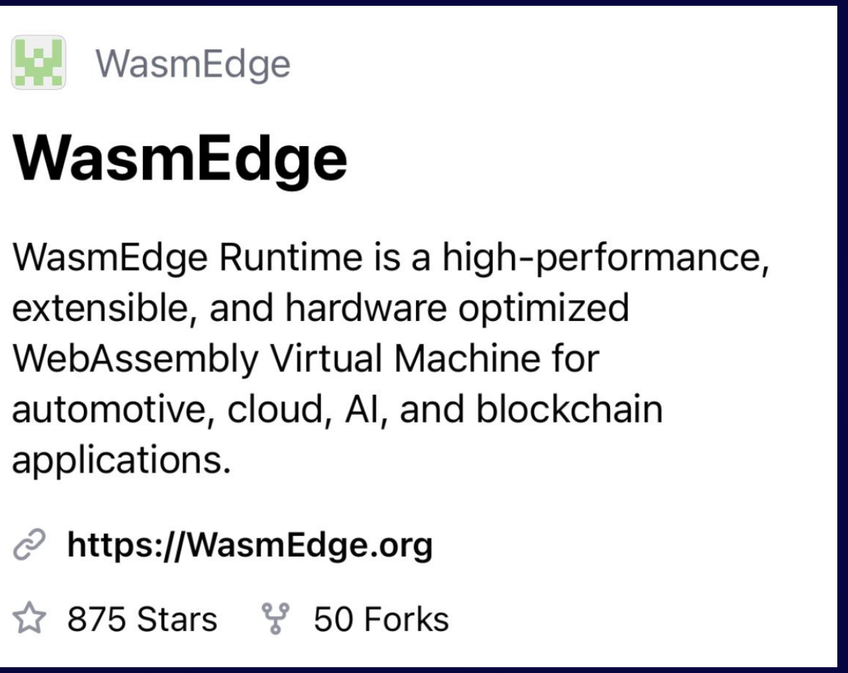
Python still popular. Visual Basic for Applications liked about as much as meetings

By Richard Speed 9 Apr 2019 at 19:00 99 SHARE



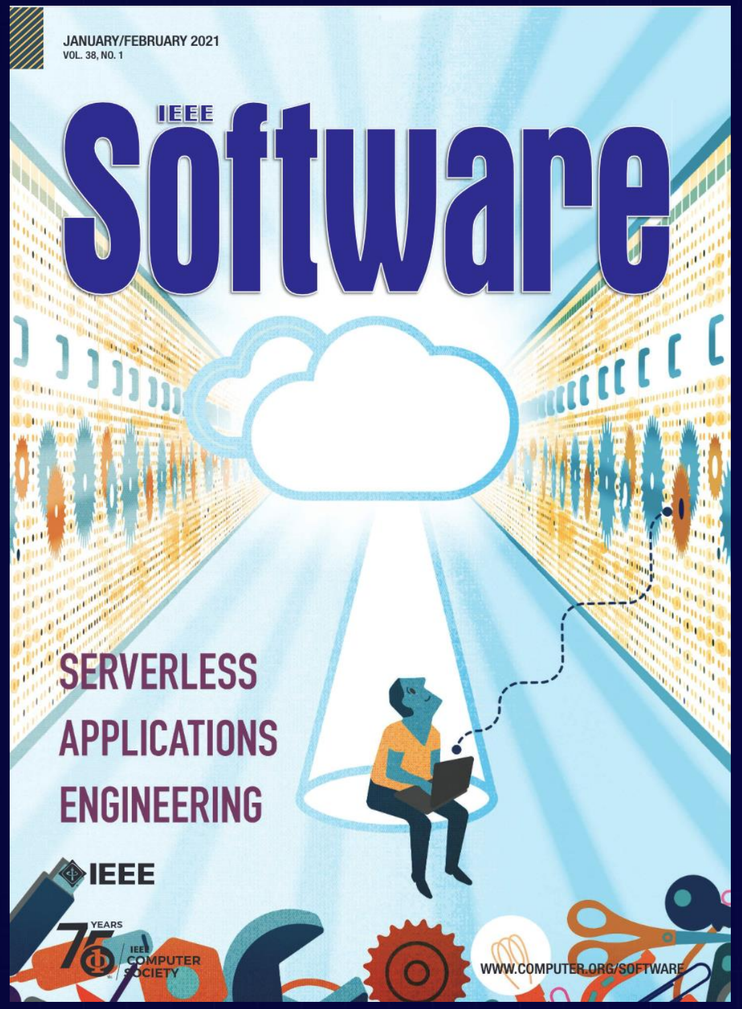
It seems coders cannot get enough of Rust, according to a survey conducted by dev saviours Stack Overflow.

The 2019 survey had almost 90,000 (a bit down on the 100,000 from 2018) developers venting their collective spleen on life, languages and loathings.



https://github.com/WasmEdge/WasmEdge

CNCF 沙箱项目



	Binary-tree	Fannkuch-redux	Mandelbrot	nbody
Azure				
Docker	16.9 ± 0.9	26.1 ± 0.1	16.2 ± 0.9	4.1 ± 0.05
Lucet	Failed	56.6 ± 0.1	Failed	4.67 ± 0.03
SSVM	12.2 ± 0.1	32.8 ± 0.2	12.7 ± 0.1	3.75 ± 0.03
V8	17.4 ± 0.1	30.0 ± 0.2	10.45 ± 0.04	3.38 ± 0.02
WAVM	13.4 ± 0.1	29.4 ± 0.1	11.8 ± 0.07	3.74 ± 0.02
AWS				
Docker	29.9 ± 0.1	39.2 ± 0.1	11.29 ± 0.09	4.14 ± 0.06
Lucet	Failed	67.8 ± 0.1	Failed	5.52 ± 0.08
SSVM	13.4 ± 0.1	38 ± 0.1	10.8 ± 0.2	3.92 ± 0.08
V8	19.2 ± 0.2	40.±0.8	10.6 ± 0.1	3.69 ± 0.08
WAVM	15.1 ± 0.1	36.5 ± 0.1	9.96 ± 0.08	3.95 ± 0.08

WasmEdge 扩展

WasmEdge 与其它的 WebAssembly 虚拟机的关键区别是它对非标准扩展的支持。WASI 规范为开发者提供了一种有效且安全地扩展 WebAssembly 虚拟机的机制。 WasmEdge 团队根据现实世界的客户需求创建了以下类似 WASI 的扩展。

- Tensorflow. 开发者可以使用 一个简单的 Rust API 编写 Tensorflow 推理函数，然后在 WasmEdge 内以本机速度安全地运行该函数。
- 其他AI框架。除了 Tensorflow，Second State 团队还在为 AI 框架（如 用于 WasmEdge 的ONNX 和 Tengine）构建类 WASI 的扩展。
- 存储。 WasmEdge 存储接口 允许 WebAssembly 程序读取和写入键值存储。
- 命令界面。WasmEdge 让 Webassembly 功能可以执行宿主机操作系统的本地命令。它支持传递参数、环境变量、STDIN/STDOUT pipes 和宿主机访问的安全策略。
- 以太坊。 WasmEdge Ewasm 扩展支持编译为 WebAssembly 的以太坊智能合约。它是以太坊风格的 WebAssembly (Ewasm) 的领先实现。
- Substrate。 Pallet 让 WasmEdge 能在任何基于 Substrate 的区块链上充当以太坊智能合约执行引擎。

https://github.com/WasmEdge/WasmEdge

```rust
pub fn infer(image_data: &[u8]) -> String {
    let img = image::load_from_memory(image_data).unwrap().to_rgb();
    let resized = image::imageops::thumbnail(&img, 192, 192);
    let mut flat_img: Vec<f32> = Vec::new();
    for rgb in resized.pixels() {
        flat_img.push(rgb[0] as f32 / 255.);
        flat_img.push(rgb[1] as f32 / 255.);
        flat_img.push(rgb[2] as f32 / 255.);
    }
```

```
let model_data: &[u8] = include_bytes!("mobilenet_v1_192res_1.0_seefood.pb");
let labels = include_str!("aiy_food_V1_labelmap.txt");

let mut session = ssvm_tensorflow_interface::Session::new(model_data,
    ssvm_tensorflow_interface::ModelType::TensorFlow);
session.add_input("input", &flat_img, &[1, 192, 192, 3])
       .add_output("MobilenetV1/Predictions/Softmax")
       .run();
let res_vec: Vec<f32> = session.get_output("MobilenetV1/Predictions/Softmax");
```

```rust
let mut i = 0;
let mut max_index: i32 = -1;
let mut max_value: f32 = -1.0;
while i < res_vec.len() {
    let cur = res_vec[i];
    if cur > max_value {
        max_value = cur;
        max_index = i as i32;
    }
    i += 1;
}
println!("{} : {}", max_index, max_value);
```

```
let mut confidence = "could be";
if max_value > 0.75 {
    confidence = "is very likely";
} else if max_value > 0.5 {
    confidence = "is likely";
} else if max_value > 0.2 {
    confidence = "could be";
} else {
    return "It does not appears to be a food item in the picture.".to_string();
}

let mut label_lines = labels.lines();
for _i in 0..max_index {
  label_lines.next();
}

let bird_name = label_lines.next().unwrap().to_string();
return format!("It {} a <a href='https://www.google.com/search?q={}'>{}</a> in the picture",
    confidence.to_string(), bird_name, bird_name);
```

代码:
https://github.com/second-state/tencent-tensorflow-scf

Demo:
http://secondstate.info/tencent/

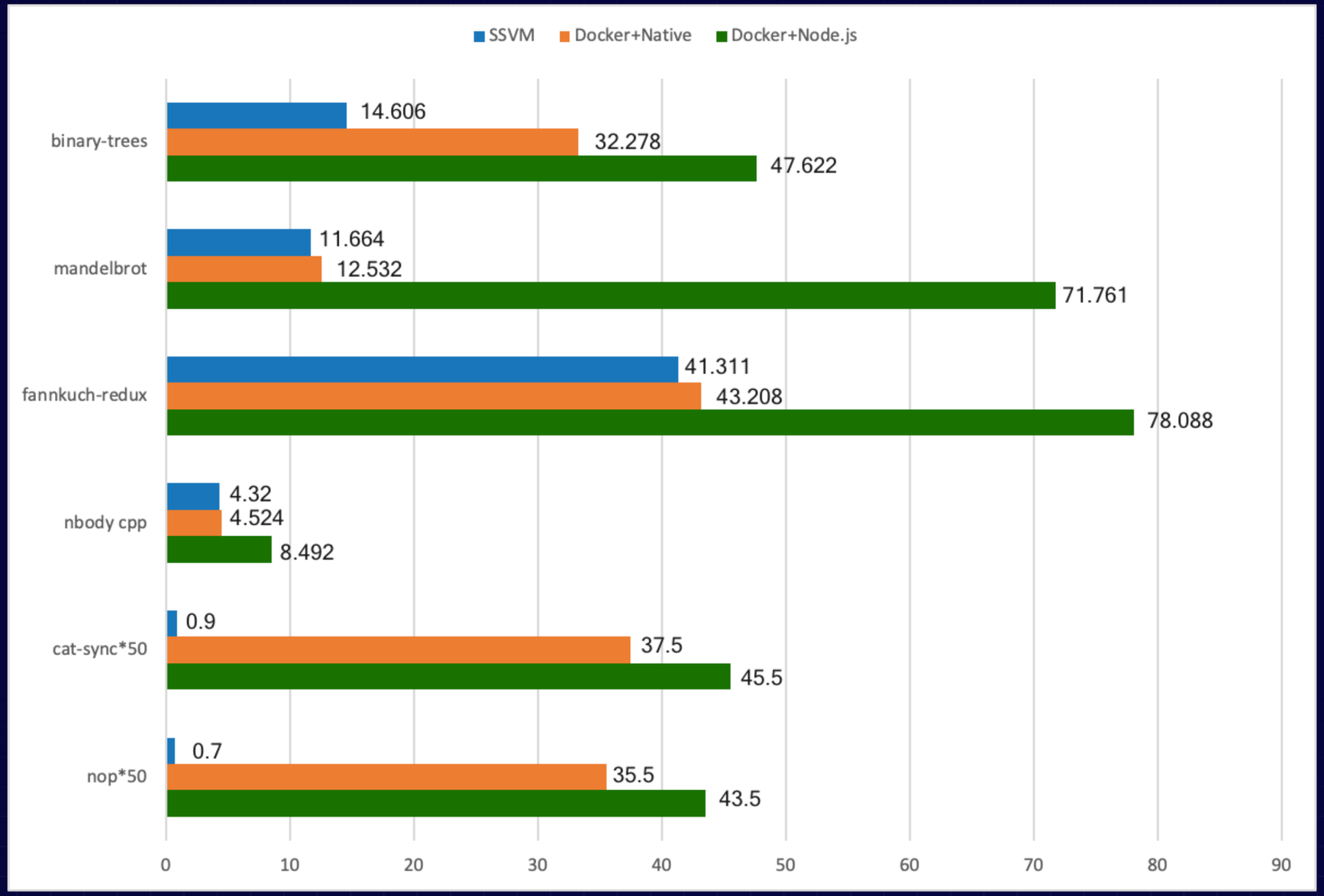
教程:
https://www.secondstate.io/articles/faas-image-classification/

扫码体验
Serverless AI 推理

WasmEdge 作为 Docker 的替代

1. Hypervisor VM and microVMs （例如 AWS Firecracker）
2. 应用容器（例如 Docker）
3. 高级语言虚拟机 （例如 JVM、Ruby / Python 运行环境、v8、WebAssembly）

Wasm 与 Docker 的对比



WebAssembly 的好处

01

安全

安全运行由第三方开发人员编写的不受信任且可能存在错误的代码

02

性能

由于动态编译器优化，接近或超过本机性能

03

轻量级

可以嵌入到其他语言或运行时框架中，在进程或线程中启动

04

可移植

适用于许多操作系统和硬件平台，包括传统和实时操作系统

WasmEdge 的优势

- 符合 WebAssembly 标准
- 支持所有提议的扩展
- 高性能
- AI 和其他的自定义扩展
- 支持旧版的 Linux (manylinux1) 和 RTOS.
- 支持 OCI 和 k8s 生态

已经支持了 cri-o

https://github.com/second-state/runw

很快将支持 k8s、 KubeEdge、KubeSphere

https://github.com/second-state/crunw

敬请期待

THANK YOU!

感谢聆听！