Get [your friends] started with Rust

SERVERLESS WASM by Second State

A Jamstack hello world

Image processing

AI inference

RUST CHINA CONF 2020

首届中国 Rust 开发者大会

2020.12.26–27 深圳

RUST CHINA CONF 2020

首届中国 Rust 开发者大会

2020.12.26–27 深圳

Rust China Conf 2020

Shenzhen, China

2020conf.rustcc.cn

Rust and WebAssembly

A match made in the cloud
云中的浪漫

Michael Yuan
michael@secondstate.io
https://www.SecondState.io/

Rust + WebAssembly ~ Java + JVM

Plenty of room at the bottom

Plenty of Room at the Bottom

Richard P. Feynman
(Dated: Dec. 1959)

This is the transcript of a talk presented by Richard P. Feynman to the American Physical Society in Pasadena on December 1959, which explores the immense possibilities afforded by miniaturization.

I imagine experimental physicists must often look with envy at men like Kamerlingh Onnes, who discovered a field like low temperature, which seems to be bottomless and in which one can go down and down. Such a man

can easily be adjusted in size as required by the photo-engraving, and there is no question that there is enough room on the head of a pin to put all of the Encyclopaedia Brittanica.

a.k.a Moore’s Law

Plenty of room at the top

REVIEW

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]

[1]Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA, USA.

[2]MIT Initiative on the Digital Economy, Cambridge, MA, USA.

[3]NVIDIA Research, Westford, MA, USA.

[4]Microsoft Research, Cambridge, MA, USA.

↵*Corresponding author. Email: neil_t@mit.edu

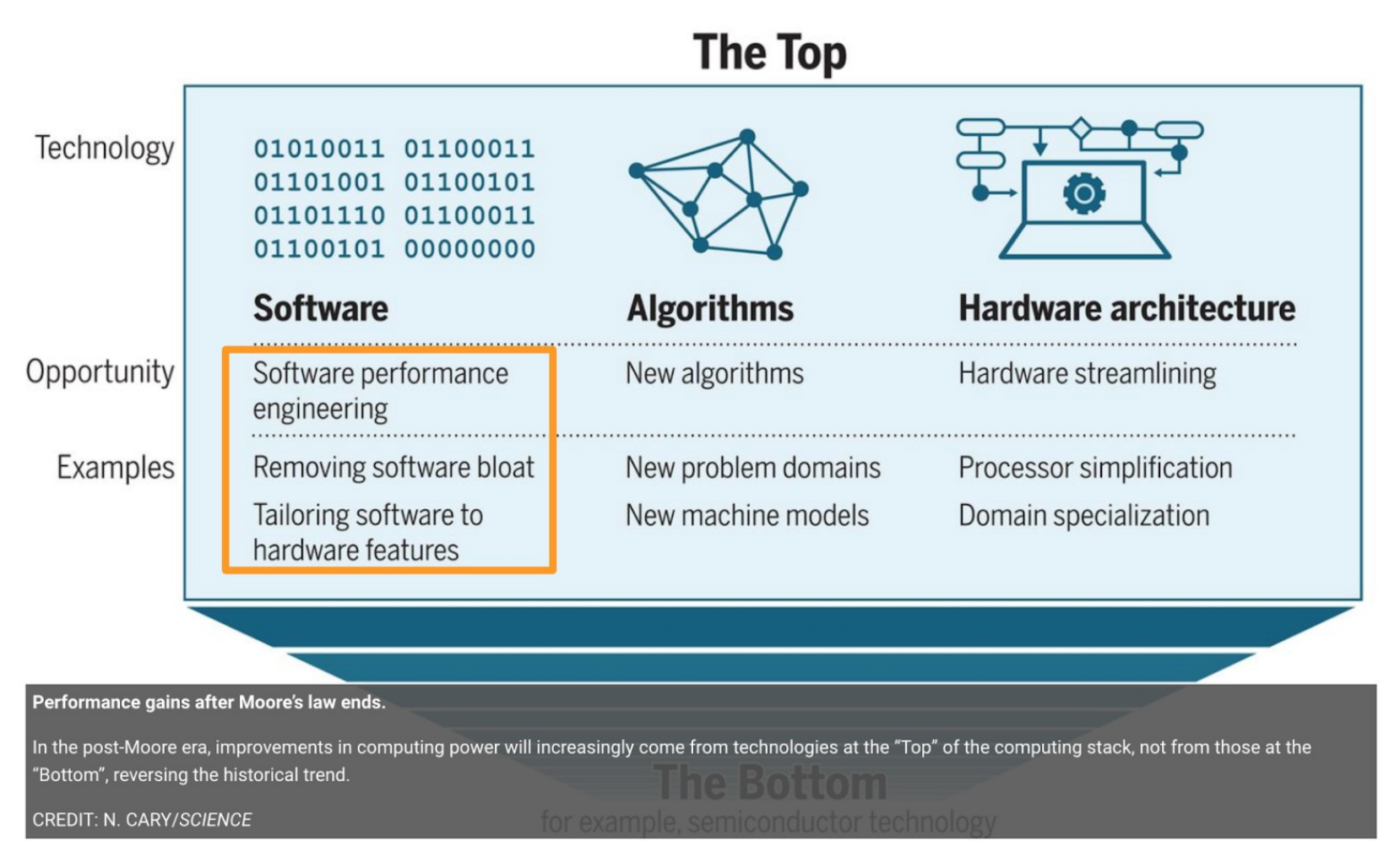
↵† Present address: Google, Cambridge, MA, USA.

– Hide authors and affiliations

Science 05 Jun 2020:
Vol. 368, Issue 6495, eaam9744
DOI: 10.1126/science.aam9744

Software performance engineering

60,000x performance gain by rewriting Python with C



Performance gains after Moore's law ends.

In the post-Moore era, improvements in computing power will increasingly come from technologies at the "Top" of the computing stack, not from those at the "Bottom", reversing the historical trend.

CREDIT: N. CARY/*SCIENCE*

Rust is highly performant

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

Developers love Rust

Report: Rust is the most beloved programming language for five years running

Latest News Published: May 29th, 2020 - Jakub Lewkowicz

Twitter Email + More

Rust continues to maintain its spot as the most beloved of programming languages among professional developers. For the firth year in a row, the Stack Overflow Developer Survey found a majority of developers who are currently using it (86%) say that they are interested in continuing to develop with it.

But, cloud native is not machine native

"I think there is a world market for maybe five computers."

Thomas Watson, president of IBM, 1943

But he was off by 4 …

We need to optimize for the world computer!

Lessons from Java

- Server-side computing has its own requirements
  - Isolation and safety
  - Security
  - Cross platform compatibility
- Programming language trend matters
  - Developer productivity
- The 10-year cycles
  - “Reinventing the wheels” vs “Don’t make me eat the elephant again”

Jamstack and serverless

The modern way to build [websites & apps] that delivers better performance

- Statically generated web pages distributed via CDN
  - Markup for UI
  - JavaScript for interactions
- APIs for the backend services
  - Serverless functions and databases

Solomon Hykes @solomonstre · Mar 28, 2019

If WASM+WASI existed in 2008, we wouldn't have needed to created Docker. That's how important it is. Webassembly on the server is the future of computing. A standardized system interface was the missing link. Let's hope WASI is up to the task!

Solomon Hykes
@solomonstre

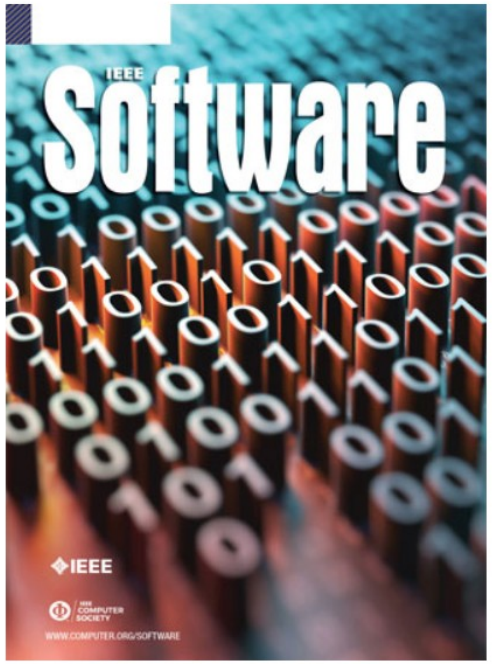
Co-founder of Docker. He/him. I follow lots of people because I have a lot to learn.

WebAssembly is more abstract than Docker

- Hypervisor VM and microVMs (e.g., AWS Firecracker)
- Application containers (e.g., Docker)
- High level language VMs (e.g., JVM, Ruby / Python runtimes, v8, WebAssembly)

WebAssembly vs Docker

- WebAssembly is 100x faster at cold start
- WebAssembly is 10% to 50% faster at execution time
  - AOT compiler could be faster than native
- WebAssembly has a much smaller footprint
- WebAssembly has a modern security model
- WebAssembly is composible
- WebAssembly integrates into popular frameworks



Download PDF

Generate Citation

Home / Magazines / IEEE Software / Preprints

A Lightweight Design for High-Performance Serverless Computing

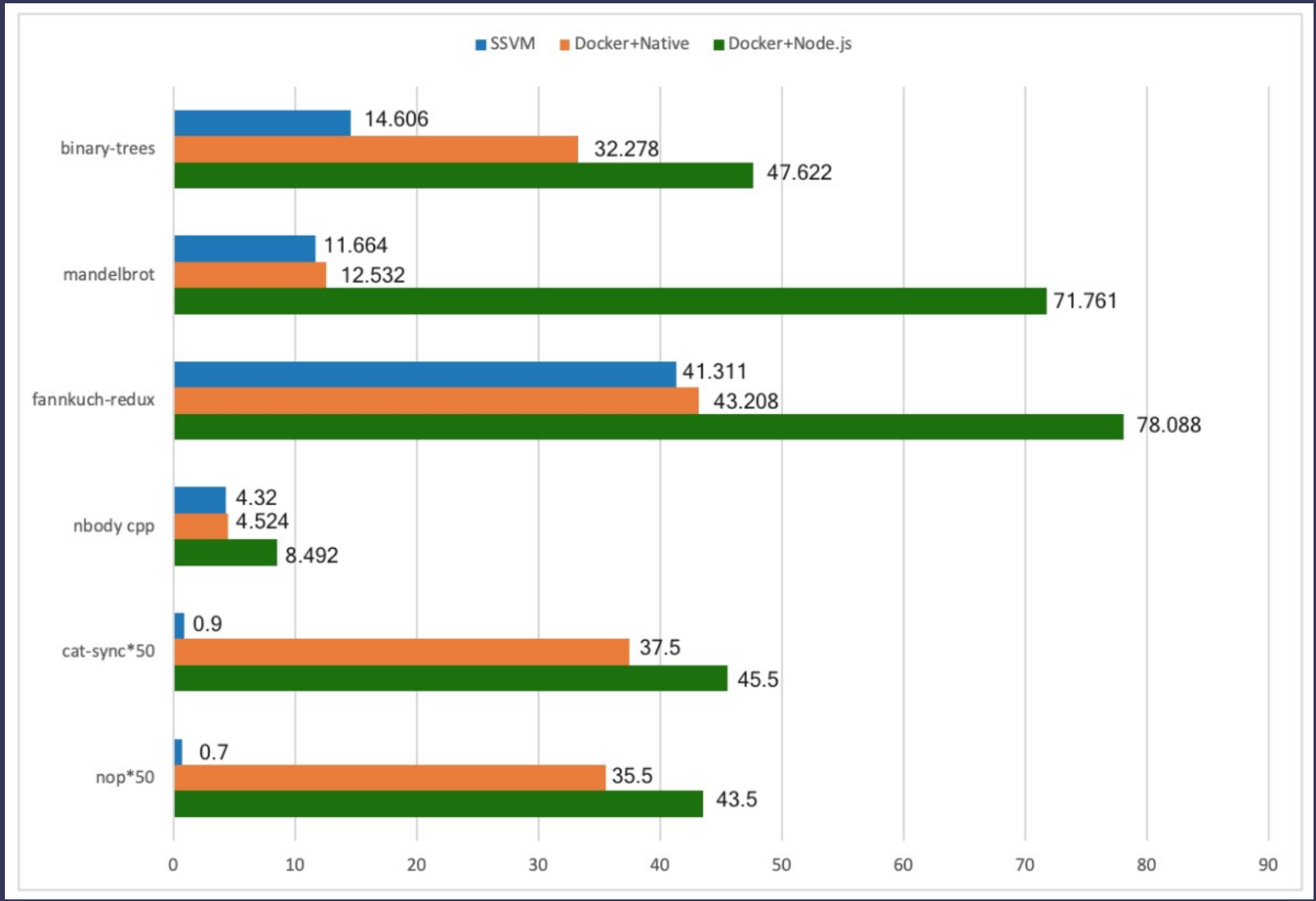
PrePrints pp. 0-0,
DOI Bookmark: 10.1109/MS.2020.3028991

Keywords

Runtime, Cloud Computing, FAA, Containers, Benchmark Testing, Servers, Virtual Machining

Authors

Ju Long, Computer Information Systems and Quantitative Methods, Texas State University, San Marcos, Texas 78666 United States
Hung-Ying Tai, R&D, Second State LLC, Austin, Texas United States
Shen-Ta Hsieh, R&D, Second State LLC, Austin, Texas United States
Juntao Michael Yuan, R&D, Second State LLC, Austin, Texas United States


SSVM
Docker+Native
Docker+Node.js
binary-trees
14.606
32.278
47.622
mandelbrot
11.664
12.532
71.761
fannkuch-redux
41.311
43.208
78.088
nbody cpp
4.32
4.524
8.492
cat-sync*50
0.9
37.5
45.5
nop*50
0.7
35.5
43.5
0
10
20
30
40
50
60
70
80
90

WebAssembly and Rust

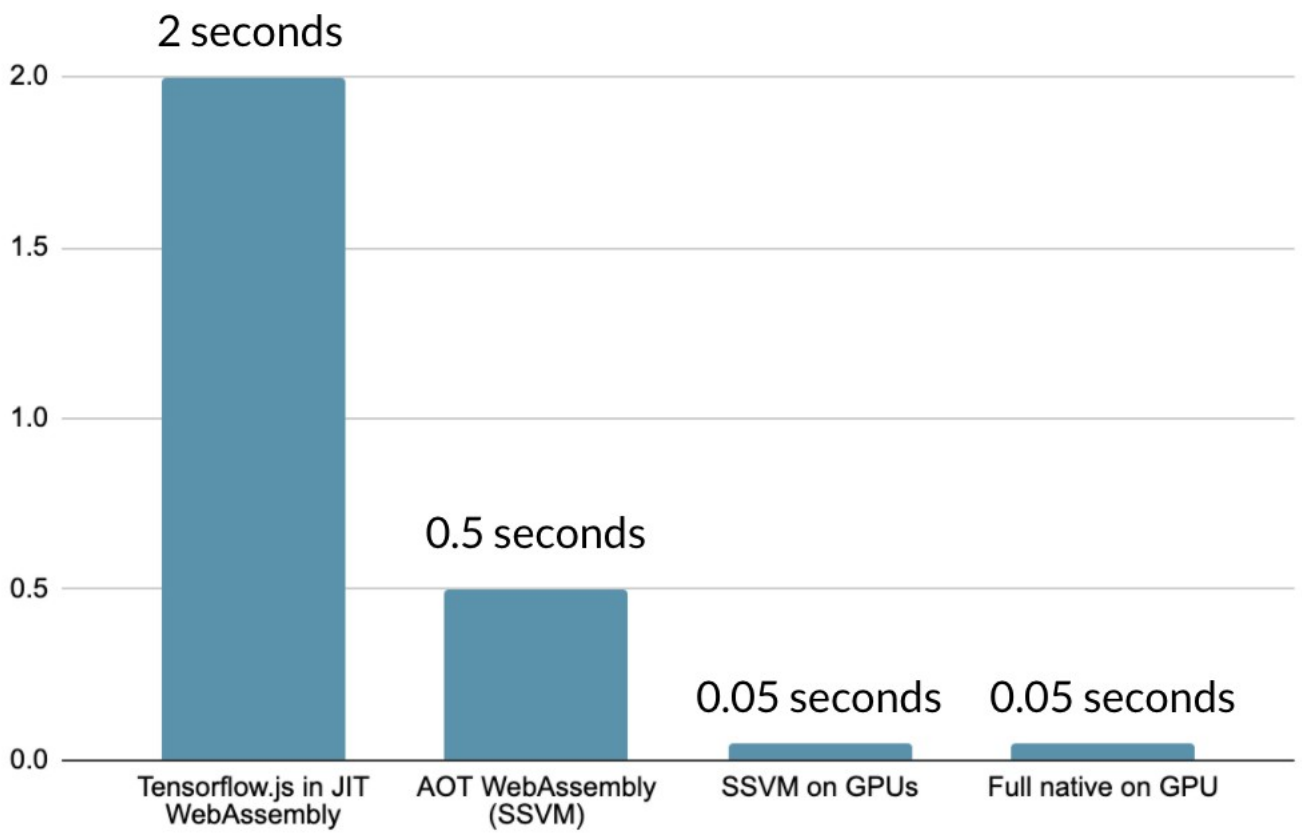
- Rust
  - Performance
  - Developer productivity
  - Memory safety
- WebAssembly
  - Performance
  - Isolation and sandbox
  - Security
  - Cross platform

WebAssembly is a first class compiler target for the Rust toolchain.

WebAssembly Systems Interface (WASI)

- Safe access to the file system, I/O, _start(), environment variables, command line arguments, and other standard library API
- WASI for sockets and network
- WASI for GPU or ASIC-enabled tensorflow (by SSVM)
- WASI for storage (by SSVM)
- WASI for precise usage metering (by SSVM)
- WASI for native commands and threads (by SSVM)
- WASI for Ethereum

A WASI-like API for Tensorflow



WASI Tensorflow for image classification

```rust
#[wasm_bindgen]
pub fn infer(image_data: &[u8]) -> String {
    let img = image::load_from_memory(image_data).unwrap().to_rgb();
    let resized = image::imageops::thumbnail(&img, 224, 224);
    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.);
    }
```

WASI Tensorflow for image classification

```
let model_data: &[u8] = include_bytes!("mobilenet_v2_1.4_224_frozen.pb");
let labels = include_str!("imagenet_slim_labels.txt");

let mut session = ssvm_tensorflow_interface::Session::new(model_data, ssvm_tensorflow_interface::ModelType::TensorFlow);
session.add_input("input", &flat_img, &[1, 224, 224, 3])
       .add_output("MobilenetV2/Predictions/Softmax")
       .run();
let res_vec: Vec<f32> = session.get_output("MobilenetV2/Predictions/Softmax");
```

WASI Tensorflow for image classification

```
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;
}
```

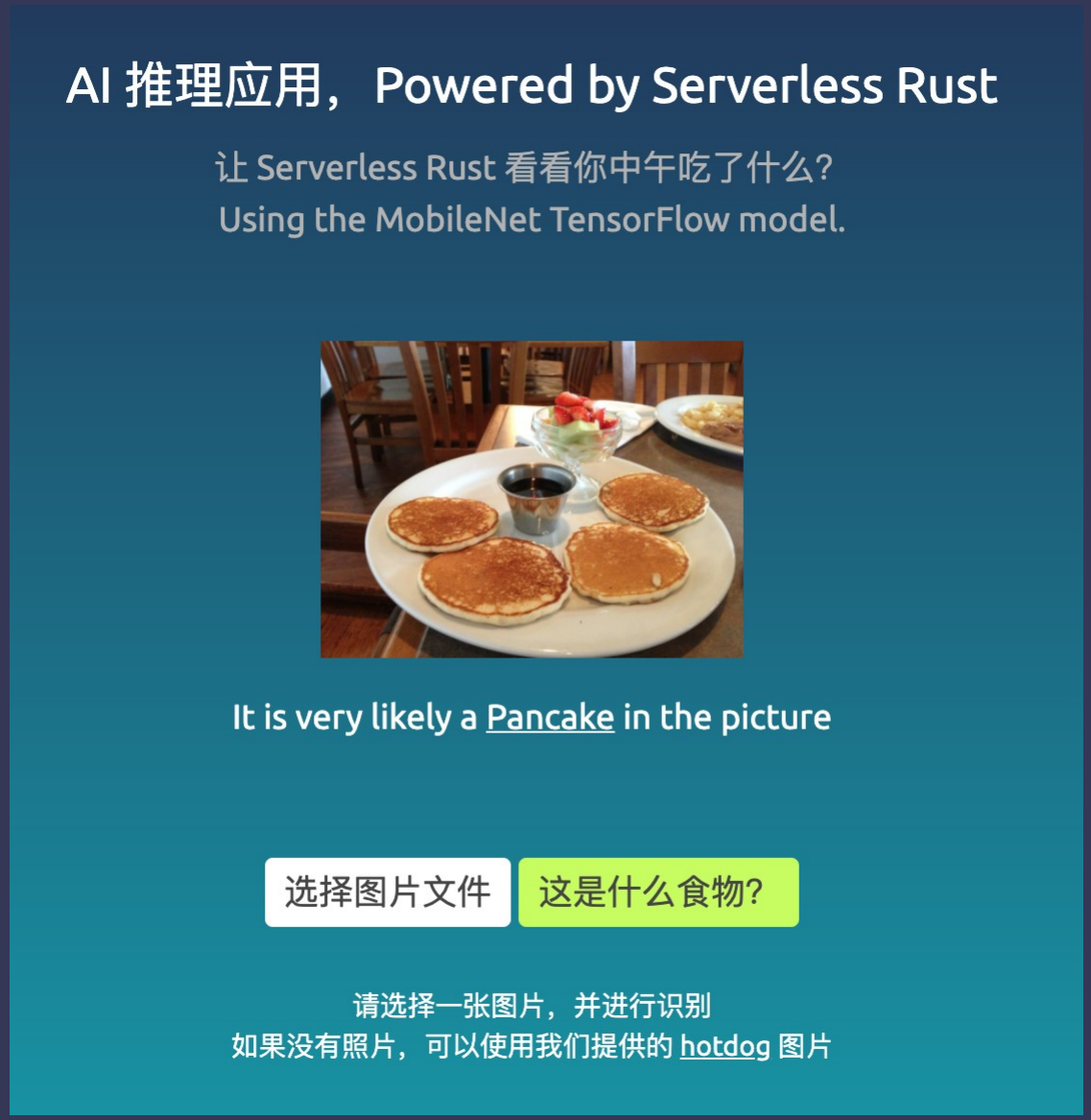
WASI Tensorflow for image classification

```
let mut confidence = "low";
if max_value > 0.75 {
    confidence = "very high";
} else if max_value > 0.5 {
    confidence = "high";
} else if max_value > 0.2 {
    confidence = "medium";
}

let mut label_lines = labels.lines();
for _i in 0..max_index {
  label_lines.next();
}
let ret: (String, String) = (label_lines.next().unwrap().to_string(), confidence.to_string());
```

Try it out!

https://www.secondstate.io/rustchinaconf/



It is your turn now!

用 Rust 写 Serverless 函数

Challenge #1 用 Rust 写一个 serverless function

Prize: Serverless Rust Swag by Second State

Challenge #2 用 Rust 写 AI as a service 云函数

Prize: Keychron K2 Wireless Mechanical Keyboard (Version 2)

Challenge #3 将 SSVM 扩展到更多应用场景与平台

Prize: Mac mini (M1 Chip)

Thank you!
https://www.SecondState.io/