5miles®

SECOND STATE

Serverless 与 AI 推理

如何在 Serverless 中使用 Wasm 进行高效 AI 推理

吕艺

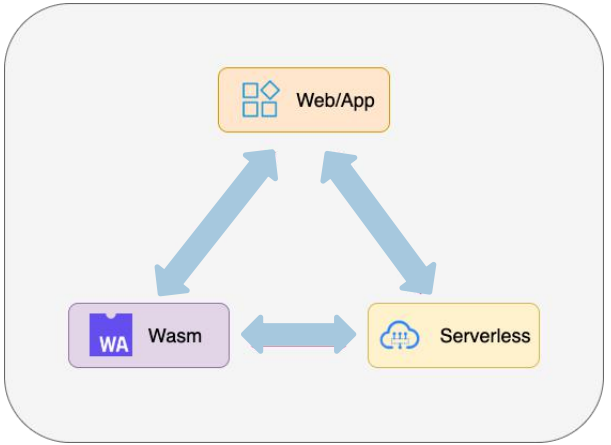
在 5miles 用 AI 做什么

我们使用 TensorFlow 模型 + WebAssembly 来检测其他平台的 logo。

- 提高效率
- 高性能
- 降低成本

目录

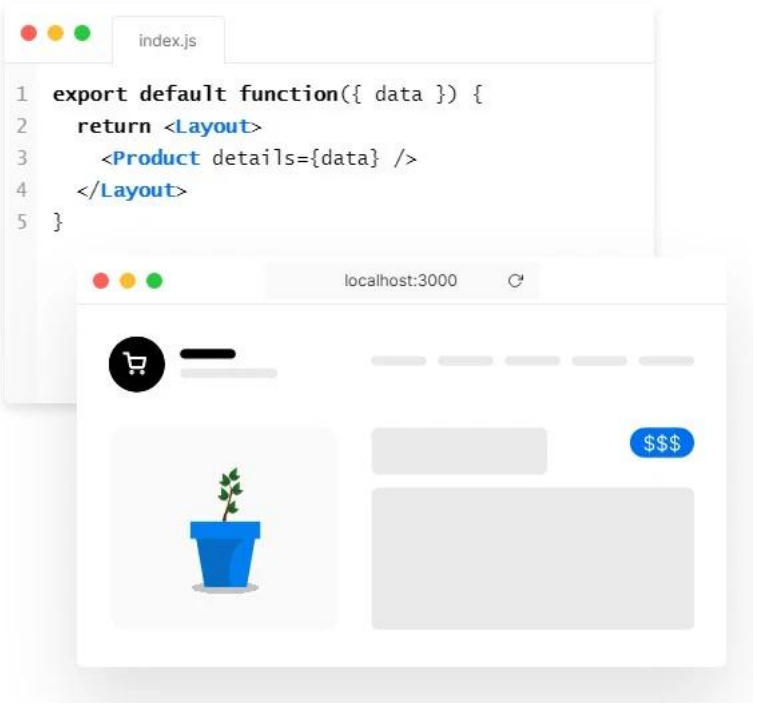
- Serverless 面临的新挑战
- 大前端的 AI 推理
- WebAssembly(Wasm) 的优势
- 用 Wasm 进行 TensorFlow 推理的方法
- Show me the code



Serverless 面临的新挑战

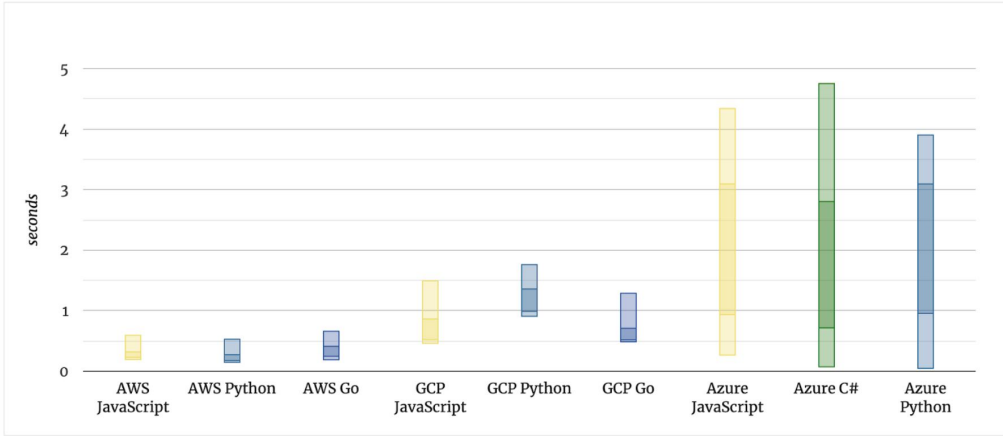
先来说说 Vercel

- “最好用”的网站托管服务
- 前端业务通过编译生成
- 支持部署云端 Serverless



对 Serverless 的要求

- 并发数量大
- 启动快
- 响应快



Typical cold start durations per language

Serverless 面临 AI 的挑战

- 模型文件大
- 处理速度要求高
- 跨平台计算能力

大前端的 AI 推理

前端推理引擎

- Web
  - TensorFlow.js（通过WebGL，WASM优化）
  - MNN.js
  - Paddle.js
  - ONNX.js
  - WebDNN

Smaller models

Model	WebGL (ms)	WASM (ms)	Plain JS (ms)	Size (MB)
BlazeFace	22.5	15.6	315.2	0.4
FaceMesh	19.3	19.2	335	2.8
Speech commands	14.7	5.3	47.3	5.6

Larger models

Model	WebGL (ms)	WASM (ms)	Plain JS (ms)	Size (MB)
PoseNet	42.5	173.9	1514.7	4.5
BodyPix	77	188.4	2683	4.6
MobileNet v2	37	94	923.6	13

tensorflow-js

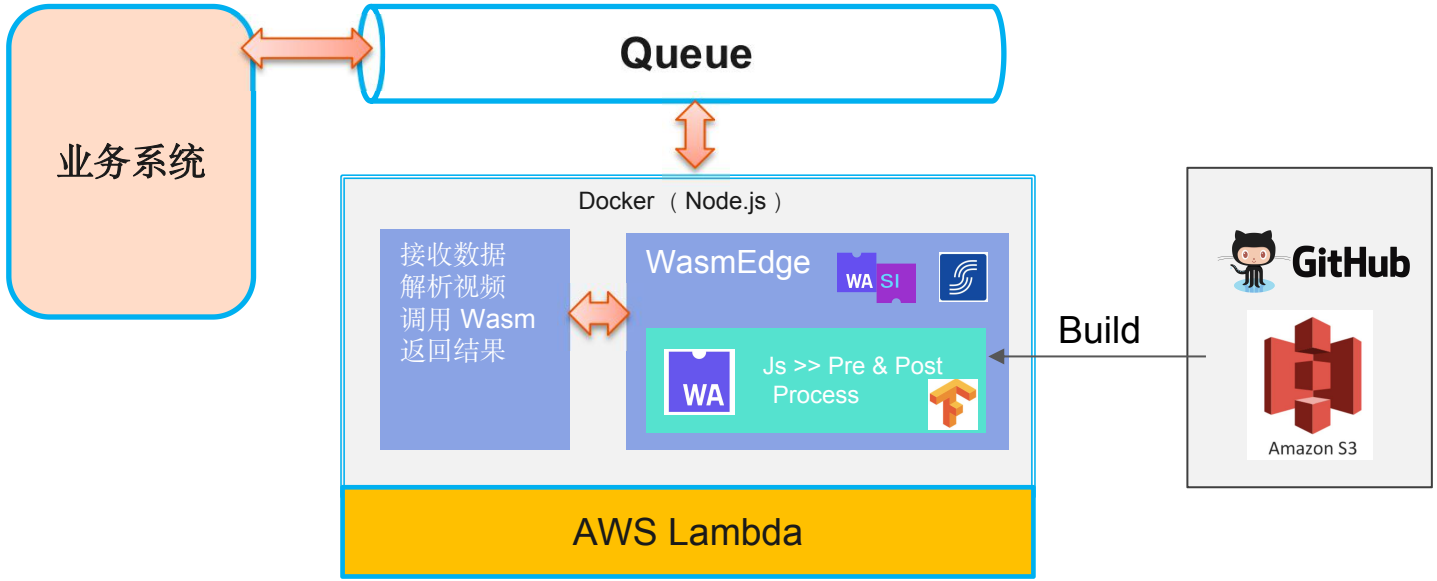
前端推理引擎

- Android / iOS
  - TensorFlow Lite　　Google
  - TNN（NCNN）　　腾讯
  - MACE　　小米
  - MNN　　阿里
  - Paddle Lite　　百度

Serverless 推理实例

- 基于腾讯云的 AI 示例
  - https://github.com/second-state/tencent-tensorflow-scf

在 Serverless 中基于 Wasm 的流式推理实践



前端推理 vs 后端推理

- 前端推理
  - 离数据近
  - 实时性好
  - 个人隐私
  - 节省流量

前端推理 vs 后端推理

- 前端推理
  - 离数据近
  - 实时性好
  - 个人隐私
  - 节省流量
- 后端推理
  - 适合较大模型
  - 业务逻辑封闭
  - 不需要实时反馈
  - 需要更好的性能

WebAssembly (Wasm) 的优势

WebAssembly (Wasm) 的优势

- 高性能
- 沙盒安全
- 基于能力的安全
- 多种开发语言
- Product-community fit

Rust + Wasm（VM） ~ Java + JVM

WasmEdge（SSVM ）

CLOUD NATIVE SANDBOX

A popular WebAssembly VM optimized for

high-performance applications

Like https://github.com/WasmEdge/WasmEdge

用 Wasm 进行 TensorFlow 推理的方法

在 CPU 上解释执行

- 用 Rust 或 C/C++ 实现的 TensorFlow 操作符
- 编译成 WebAssembly
- 在 CPU 上以解释器模式执行
- 图像识别任务在现代 Intel CPU 上需要 10 分钟以上

Rust Tract crate
https://crates.io/crates/tract

Just-In-Time (JIT)

- 在 WebAssembly 中实现 TensorFlow 运算符
- 在 v8 等 JIT 运行时中执行程序
- 图像识别任务需要 2-3 秒

Tensorflow.js
https://github.com/tensorflow/tfjs

把模型编译成 WebAssembly

- 将 AI 模型直接编译成 WebAssembly 字节码程序
- 节省了在 runtime 将 TensorFlow 操作映射到 Wasm 操作所需的时间
- 但是很难围绕模型推理进行前后处理
- 图像识别任务需要 1-2 秒

TVM
https://github.com/apache/tvm

将 ONNX 编译成 Wasm
https://github.com/ONNC/onnc-wasm

Host function(WASI)

- 从 Wasm 调用原生的 TensorFlow 库
  - 提供了调用 TensorFlow 的 Rust API
- Rust 编译的代码围绕 TensorFlow 调用进行数据准备
- 图片识别任务在 CPU 上只要 0.5 秒

WasmEdge on Tencent Serverless
https://github.com/second-state/tencent-tensorflow-scf

AOT 与 GPU

- 使用 AOT 优化 Wasm 中的数据预处理的调用
- 使用 WASI 在 GPU 上运行 TensorFlow 推理
- 为特殊的硬件运行页数的 AI 库
  - 如. AWS 的 Inferentia 芯片
- 图片识别任务只需花费 0.05 秒

WasmEdge on Tencent Serverless
https://github.com/second-state/tencent-tensorflow-scf

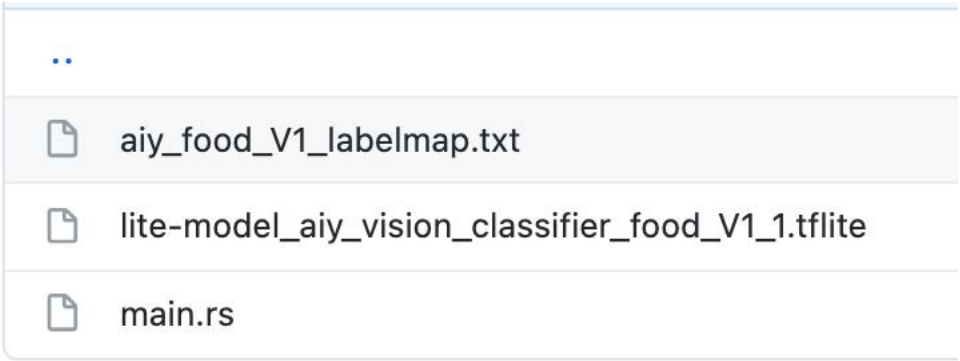
Show me the code

Serverless AI 推理的Demo

Serverless AI 推理的Demo

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

Demo: https://sls-website-ap-hongkong-kfdilz-1302315972.cos-website.ap-hongkong.myqcloud.com/



Serverless 端代码

```
// 加载模型和标签文件
let model_data: &[u8] = include_bytes!("lite-model_aiy_vision_classifier_food_V1_1.tflite");
let labels : &str = include_str!("aiy_food_V1_labelmap.txt");

// 读取图片链接
let mut buffer = String::new();
io::stdin().read_to_string(&mut buffer).expect("Error reading from STDIN");
let obj: FaasInput = serde_json::from_str(&buffer).unwrap();
let img_buf = base64::decode_config(&(obj.body), base64::STANDARD).unwrap();
```

Serverless 端代码

```
// 加载图片数据
let flat_img = ssvm_tensorflow_interface::load_jpg_image_to_rgb8(&img_buf, 192, 192);

// 实例化推理的对象
let mut session = ssvm_tensorflow_interface::Session::new(&model_data,
                                                          ssvm_tensorflow_interface::ModelType::TensorFlowLite);
// 执行推理
session.add_input("input", &flat_img, &[1, 192, 192, 3])
       .run();
// 获取推理结果
let res_vec: Vec<u8> = session.get_output("MobilenetV1/Predictions/Softmax");
```

Serverless 端代码

```rust
// 转换推理结果并输出
let mut i :i32 = 0;
let mut max_index: i32 = -1;
let mut max_value: u8 = 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;
}

//
let mut confidence :&str = "可能有";
if max_value > 200 {
    confidence = "非常可能有";
} else if max_value > 125 {
    confidence = "很可能有";
} else if max_value > 50 {
    confidence = "可能有";
}
```

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

let class_name = label_lines.next().unwrap().to_string();
if max_value > 50 && max_index != 0 {
  println!("上传的图片里面{} <a href='https://www.google.com/search?q={}'>{}</a>",
           confidence.to_string(), class_name, class_name);
} else {
  println!("上传的图片里面没有检测到食品");
}
```

Serverless 部署

- 在 Codespaces 打开 Terminal 窗口, 然后运行下面的命令行来创建云函数。

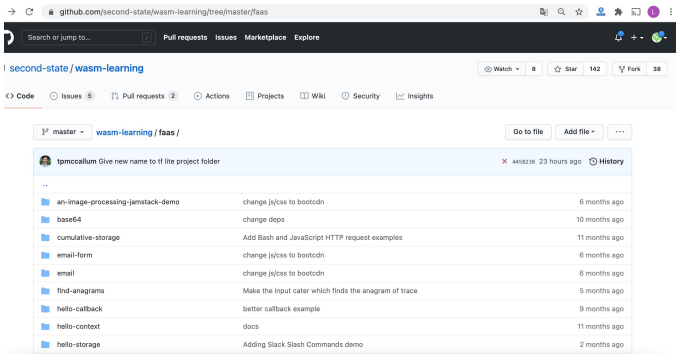
```
$ rustwasmc build --enable-aot
```

- 在 Terminal 窗口，运行下面的命令行将 TensorFlow 云函数部署到腾讯云上。

```
$ cp pkg/scf.so scf/
$ sls deploy
```

More samples

- Golang sample
  - https://github.com/second-state/WasmEdge-go-examples/tree/master/go_MobilenetFood
- fass samples
  - https://github.com/second-state/wasm-learning/tree/master/faas



展望

- 试验中的项目
  - Krustlet （https://github.com/deislabs/krustlet）
  - containerd-shim-wasm （https://github.com/dippynark/containerd-shim-wasm）
- WasmEdge
  - WasmEdge 正在逐步支持 OCI (Open Container Initiative)规范，该规范将允许由 Cloud Native Orchestration 工具，如 Kubernetes，管理 SSVM 实例。
    - WasmEdge 已经支持了 cri-o
    - 很快将支持 K8s

InfoQ PRO

关注 InfoQ Pro 服务号

你将获得：

- InfoQ 技术大会讲师 PPT 分享
- 最新全球 IT 要闻
- 一线专家实操技术案例
- InfoQ 精选课程及活动
- 定期粉丝专属福利活动