构造一个跨语言跨平台的新生态

Component Model of WebAssembly

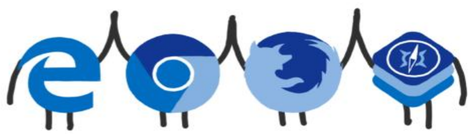
He, Liang Intel

议题

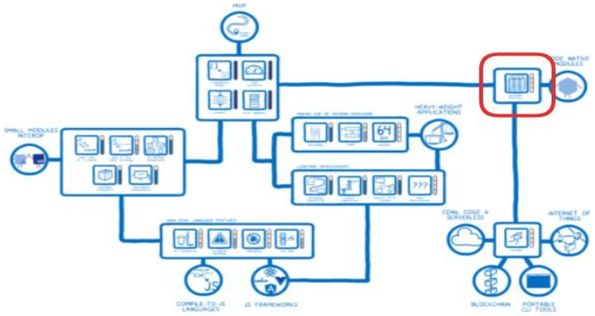
- Component Model 解决的问题
- Component Model 方案介绍
- 新 WASI

Component Model 解决的问题

回顾



WebAssembly 1.0
2017



WebAssembly Post-MVP
2018

multi-value
reference-types
tail calls
atomics

memory 64
exception handling
GC
stack-switching

WebAssembly 2.x
Now

2019

Nanoprocess vision

Wasm App 开发流程

- 使用开发者熟悉的语言编写
- 使用工具链生成 Wasm Module
- 完成胶水代码（Glue code）。包括 Runtime 初始化，Module 加载，实例化，运行

当 Wasm App 规模变大

- 模块化
  - 分离通用功能
  - 基础库（WASI）。基于能力的安全模型的潜在要求

可复用的 Wasm Module

- 实现 Wasm Modules 时：
  - 如何使用不同语言的类型系统表达相同的逻辑概念，比如复合类型，语言特有的类型（Optional），编程的概念（Async）
  - 如何确保经过不同的工具链的转换，解决了第一个问题的代码能产生符合 import/export 要求的 Wasm 代码
- 运行 Wasm Modules 时：
  - 胶水代码。定义实例化顺序，满足实例化条件（import）
  - 链接（link）的形态

链接形态

- 静态
  - 自包含但扩大了不必要的冗余
- 动态
  - Shared-everything。模拟 native linking （.dll, .so）的过程，所有的 modules 共享一个内存空间
  - Shared-nothing。充分隔离
- 根据场景制定最适合的方案

Component Model 方案介绍

Component Model 解决什么问题

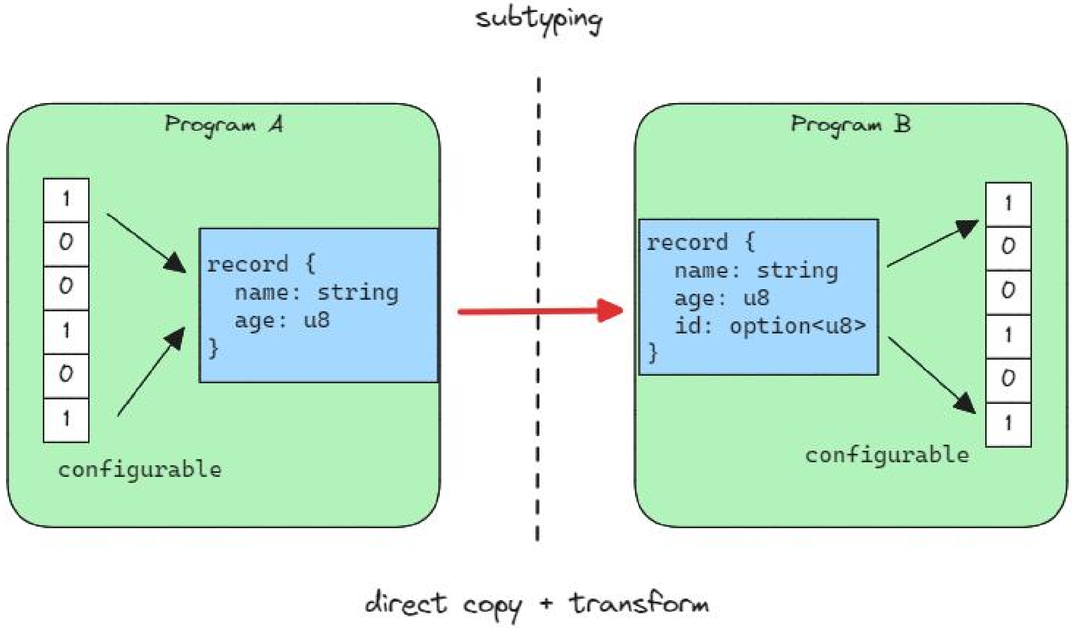
- 实现跨语言条件下的跨模块的通讯
  - Host <–> Module
  - Module <–> Module
- 实现多 Module 之间的隔离性，并使其可配置

Component type

- IDL。 .WIT
  - 丰富的类型。record,list,tuple,option,resource 等
  - 需要由 Core Wasm Type 来表示
- Canonical ABI. 实现 Component Type 和 Core Wasm Type 之间的转换
- Bindgen (Optional)。在 Canonical ABI 的约束下，实现不同语言对 Component type 的表示

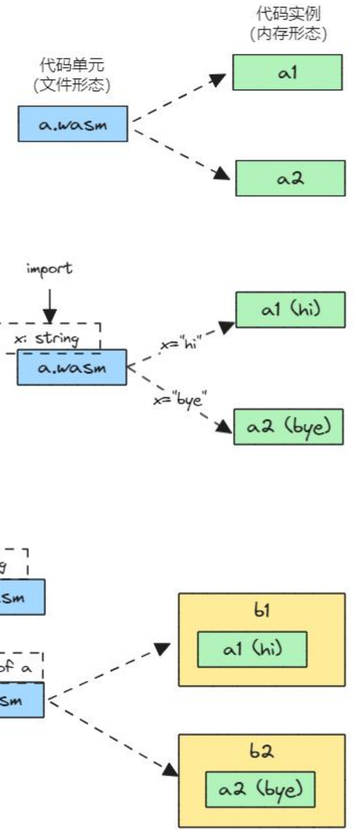
基于 component type 的接口

- 参数和返回值需符合~~一致性~~兼容性原则
- 参数和返回值按值传递
- Runtime 负责在不同 component 间搬运数据



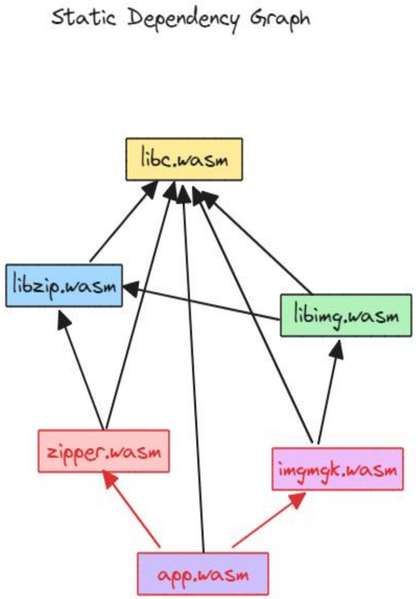
Parametric linking

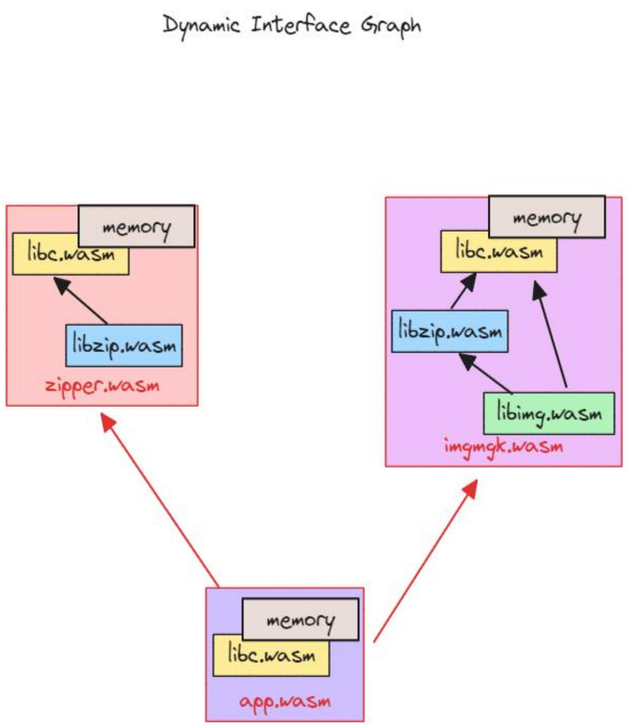
- 代码单元（静态）和代码实例（动态）
- 代码单元声明的 import 必须在实例化过程中满足
- 一个代码单元可以成为另一个代码单元的 import
- 优势：隔离和虚拟化



可配置的链接形式

- 举例：
  - component instance 之间不共享线性空间
  - component instance 内部共享线性空间





简化胶水代码

- 将大部分实例化过程（完成链接）从胶水代码中转移到 component model section 中
  - (instance), (instantiate)
  - dummy module。重新组合来自于不同 Module instance 和 host 的 export
- 将负责类型转换的 adapter function的转移到 component model 和 runtime 中
  - Canonical ABI 完成 component type 和 wasm core type 的转换
  - bindgen 完成 component type 和语言类型的转换

以 IDL 为中心的开发模式

- 基于 WIT 定义 Component 间的接口
- 使用 bind-gen 生成模板代码
- 填补模板代码，完成开发
- 利用现有工具链生成 Wasm Module
- 利用新工具将 Wasm Module 转换为 Component

新 WASI

现有 WASI preview1 的问题

- 需要模块化。Wasi-core, Wasi-socket, Wasi-threads, Wasi-NN
- 文件描述符的表示。兼容多语言的I/O
- 需要对 lists 和 string 的支持
- 非 C 语言实现 WASI 的需求

下一个版本 WASI Preview2

- 提供丰富的类型系统。摆脱类C的描述。使用了 component type
- 基于 preopen 的思路，重新定义文件操作的权限
- 功能扩展 time-zones, file locking
- 使用 resource 来支持系统资源和其他不方便拷贝的资源
- 于 Preview3 中，支持 stream 和 async
- 支持 POSIX 以外的功能
  - Wasi-key-value-store， Wasi-sql，Wasi-http-proxy

Roadmap

- Preview2：完成 WITX 到 WIT 的切换。也就是类型系统的变更 (Sep. 7th 投票)
- Preview3：提供完整的 async 支持 (没有具体时间)
- WASI 1.0：完成标准化 (没有具体时间)
- Preview2 和 preview3 或引入 breaking changes。 WASI 1.0 将平滑过渡

工具链支持

- Wit-bindgen。 https://github.com/bytecodealliance/wit-bindgen 根据 WIT 生成 C, Rust, Go, Java 的模板代码。其余语言还在开发中
- Wasm-tools。 https://github.com/bytecodealliance/wasm-tools 将 wit 和 core module 打包为 component
- Wasi_snapshot_preview1.wasm。 https://github.com/bytecodealliance/preview2-prototyping/tree/latest/crates/wasi-preview1-component-adapter 桥接使用 preview1 接口的 wasm 到 preview2 的 component

Reference

- Component model design and specification at https://github.com/WebAssembly/component-model
- Wasi-roadmap by gohman at https://github.com/WebAssembly/meetings/blob/main/wasi/2023/presentations/2023-02-09-gohman-wasi-roadmap.pdf
- Lin Clark on the WebAssembly Component Model by Lin Clark at https://www.infoq.com/podcasts/web-assembly-component-model/?utm_source=soundcloud&utm_medium=link&utm_campaign=podcast
- The path of component model by Luke Wagner at https://www.youtube.com/watch?v=phodPLY8zNE

THANKS

软件正在重新定义世界

Software Is Redefining The World