WebAssembly in ByteDance

- Wilson Wang/Varun Gupta

Introduction

Who We Are?

ByteDance Infra Lab Compute Group

The **Infrastructure System Lab** works on cutting edge infrastructure system innovations, including but not limited to **compute, storage, database, networking**, etc.

Compute Group Research Areas

- Cloud-native & Serverless
- Machine Learning
- OS & Virtualization

Internal WASM Collaborators

ByteFaas Team

- High density WASM function deployment
- Extremely short WASM code start time vs LXC.

RPC Team

- Maintaining internal WASM Runtime
- Asynchronous Hostcall Support in WASM Runtime
- WASM Custom logic as part of the mesh traffic proxy

Client-Infra Team

- Generic VM managed framework for client engineers
    - Usage example: perfect display effect(fireworks effect, etc) in mobile apps
- Internal QuickJS development that can run on WASM

WebAssembly Micro-Services

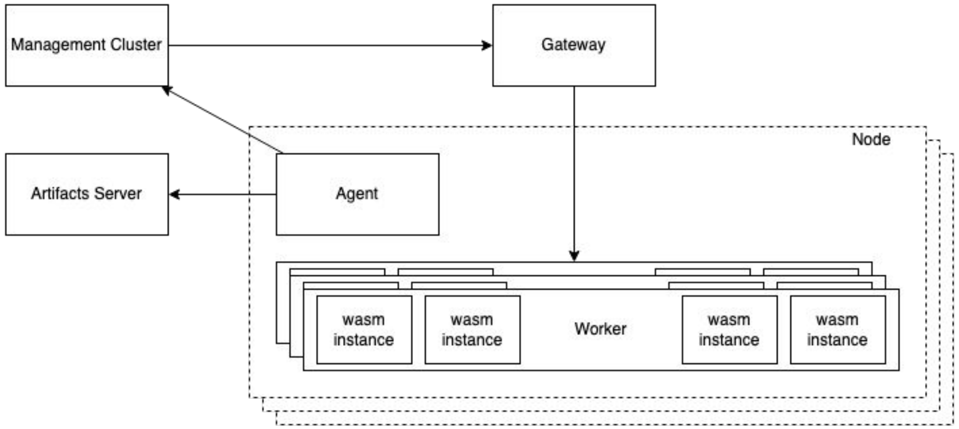
WASM Micro-Services Advantages

- Small artifacts size
  - Usually a few MBs.
- High density deployments
  - Shared memory space
- Fast loading & startup
  - A few milliseconds or sub-millisecond level.
- Secure
  - Per function per wasm runtime

Limitations? We will talk about it in the end.

Architecture

- WASM instances are deployed inside workers. Each worker can hold multiple instances.
- Other Faas components:
  - Agent is responsible for worker registration and artifacts downloading
  - Gateway is responsible for serving requests
  - Management Cluster is an HA cluster holding cluster information



Current Status

Runtime In Use

- Wasmtime (Rust based, secure)
- WasmEdge (Actively evaluating as more features supported)

Languages Support

- JavaScript (WIZER + QuickJS/SpiderMonkey)
- Rust
- C/C++
- Go (TinyGo)

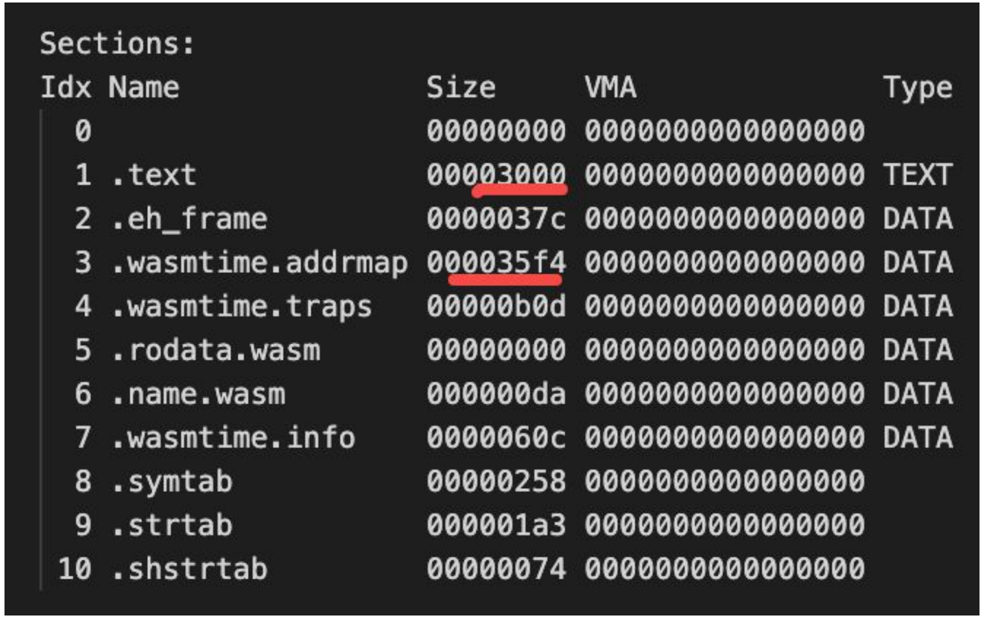
Frameworks(Ported to WASM)

- Kitex
- Hertz

Challenges & Solutions

Artifact Size(Wasmtime)

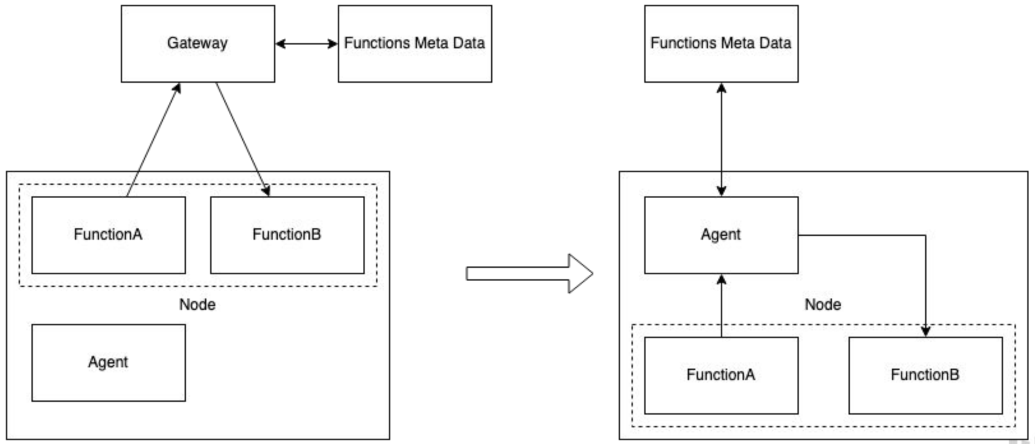
- Use Wizer to pre-initialize the WebAssembly code for a faster startup. (*.wasm to *.wasm with updated rodata section)
- Pre-compile *.wasm to native AOT code to improve performance.
    - Cranelift runs faster than llvm
- Debug information is taking a lot of space.
    - Can take around ⅓ of the total artifact size.
- **Lesson learned**: Remove unnecessary debug information in production environment.



```
Sections:
Idx Name                Size     VMA              Type
  0                     00000000 0000000000000000
  1 .text               00003000 0000000000000000 TEXT
  2 .eh_frame           0000037c 0000000000000000 DATA
  3 .wasmtime.addrmap   000035f4 0000000000000000 DATA
  4 .wasmtime.traps     00000b0d 0000000000000000 DATA
  5 .rodata.wasm        00000000 0000000000000000 DATA
  6 .name.wasm          000000da 0000000000000000 DATA
  7 .wasmtime.info      0000060c 0000000000000000 DATA
  8 .symtab             00000258 0000000000000000
  9 .strtab             000001a3 0000000000000000
 10 .shstrtab           00000074 0000000000000000
```

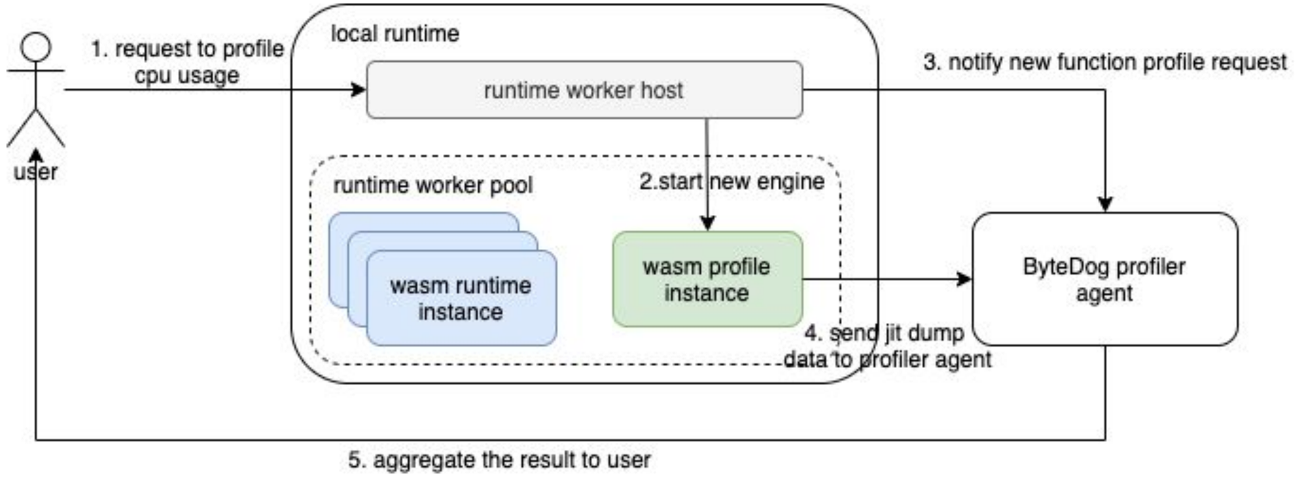
Combined Deployments

- Inter-function communication is common: Extra overhead in kernel-user level data copy, network transfer & gateway overhead.
- Solution: Investigate the function call DAG and bring up dependencies in the same worker process if possible.

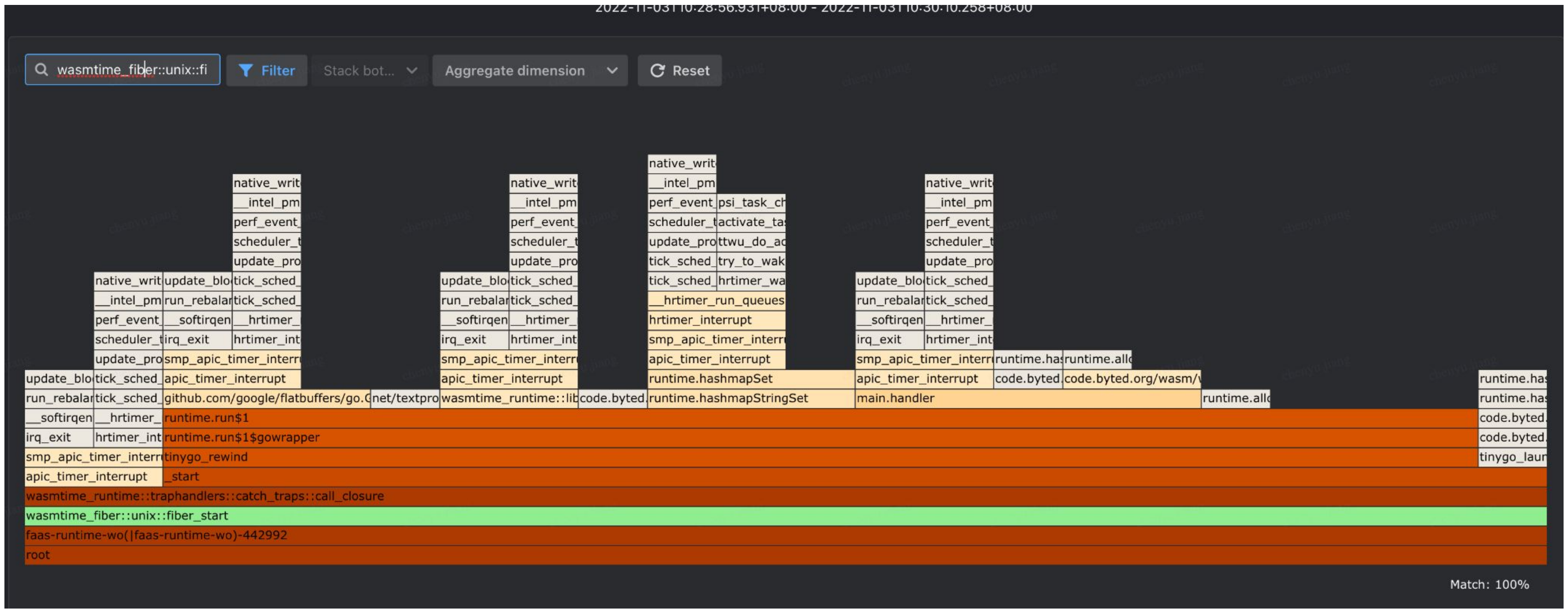


Debuggability

- Limitation of profiling - all modules are profiled for the engine, creates interference with the other running functions and bloats JITDump.
- We optimize the workflow by spawning new wasm engine to profile the function:


user
1. request to profile cpu usage
local runtime
runtime worker host
3. notify new function profile request
runtime worker pool
2.start new engine
wasm runtime instance
wasm profile instance
ByteDog profiler agent
4. send jit dump data to profiler agent
5. aggregate the result to user

Debuggability



Shared Memory

- Number of serialization and deserialization reduces by 50% from 8 to 4 counts.

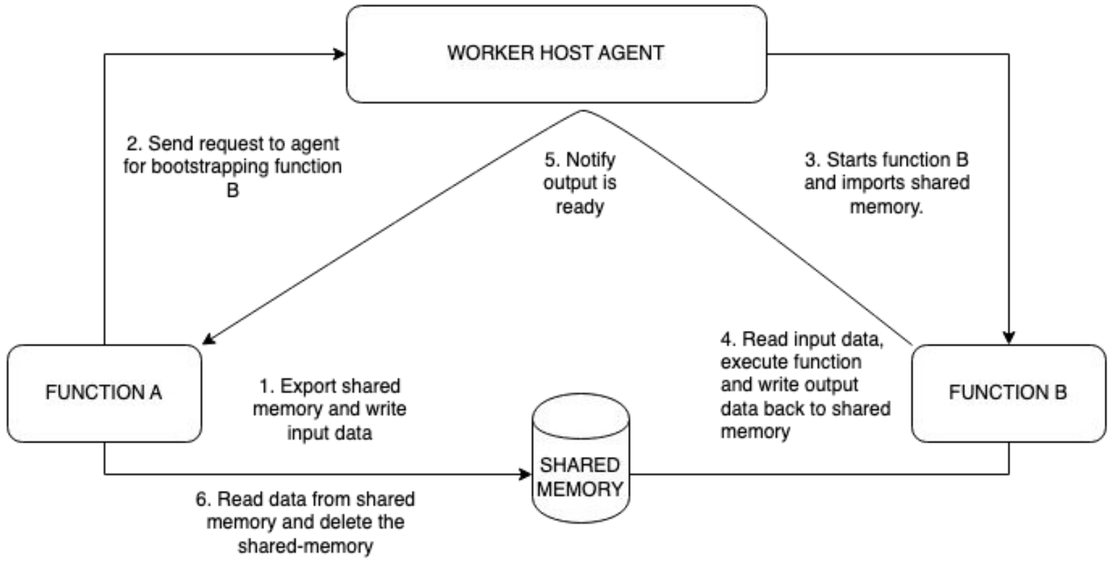
- Payload sharing happens in userspace so no data copy in Kernel space.

- No data copy or transfer via host agent.

- No interrupt or context switch between user and kernel execution.

- Scope for future improvement is to improve notification mechanism.

- Explore global memory pool to be shared by all functions rather than function pairs to reduce garbage collection.


WORKER HOST AGENT
2. Send request to agent for bootstrapping function B
5. Notify output is ready
3. Starts function B and imports shared memory.
4. Read input data, execute function and write output data back to shared memory
FUNCTION A
1. Export shared memory and write input data
SHARED MEMORY
FUNCTION B
6. Read data from shared memory and delete the shared-memory

WebAssembly Current Limitations

Limitations

- Limited language support in WebAssembly
  - Lack of language features: Coroutine, GC
  - Open-source packages support (ex golang net/http package)
- Low-level OS features support (socket, for example)
  - WasmEdge did a good job here!
- Debuggability
- GPU support
- Performance Gap (Compare to running code on native language runtime, especially interpreted languages)

We are Hiring

FTE: Cloud/Serverless Software Engineer, Researcher

Research Intern (Infrastructure System Lab)- 2023 Start (PhD):

- Seattle
  https://careers.tiktok.com/position/7154105328891119903/detail
- MTV
  https://careers.tiktok.com/position/7154105328891447583/detail

Thank You!

ByteDance 字节跳动