WebAssembly Plugin for Apache Traffic Server

Shu Kit Chan

COMMUNITY OVER CODE

Bratislava, Slovakia. June 3-5, 2024

Who? What?

Kit Chan

(kichan@apache.org / kichan@yahooinc.com)

- 19 years in Yahoo
- Software Architect in Edge/CDN Team
- Volunteer in Yahoo OSPO

- Apache Traffic Server PMC/Committer
- Wasm, Lua, ESI plugins

- WebAssembly + ATS == Framework to build functionality on your edge!

COMMUNITY OVER CODE

Apache Traffic Server

ATS & Yahoo

DDOS protection

Privacy Control

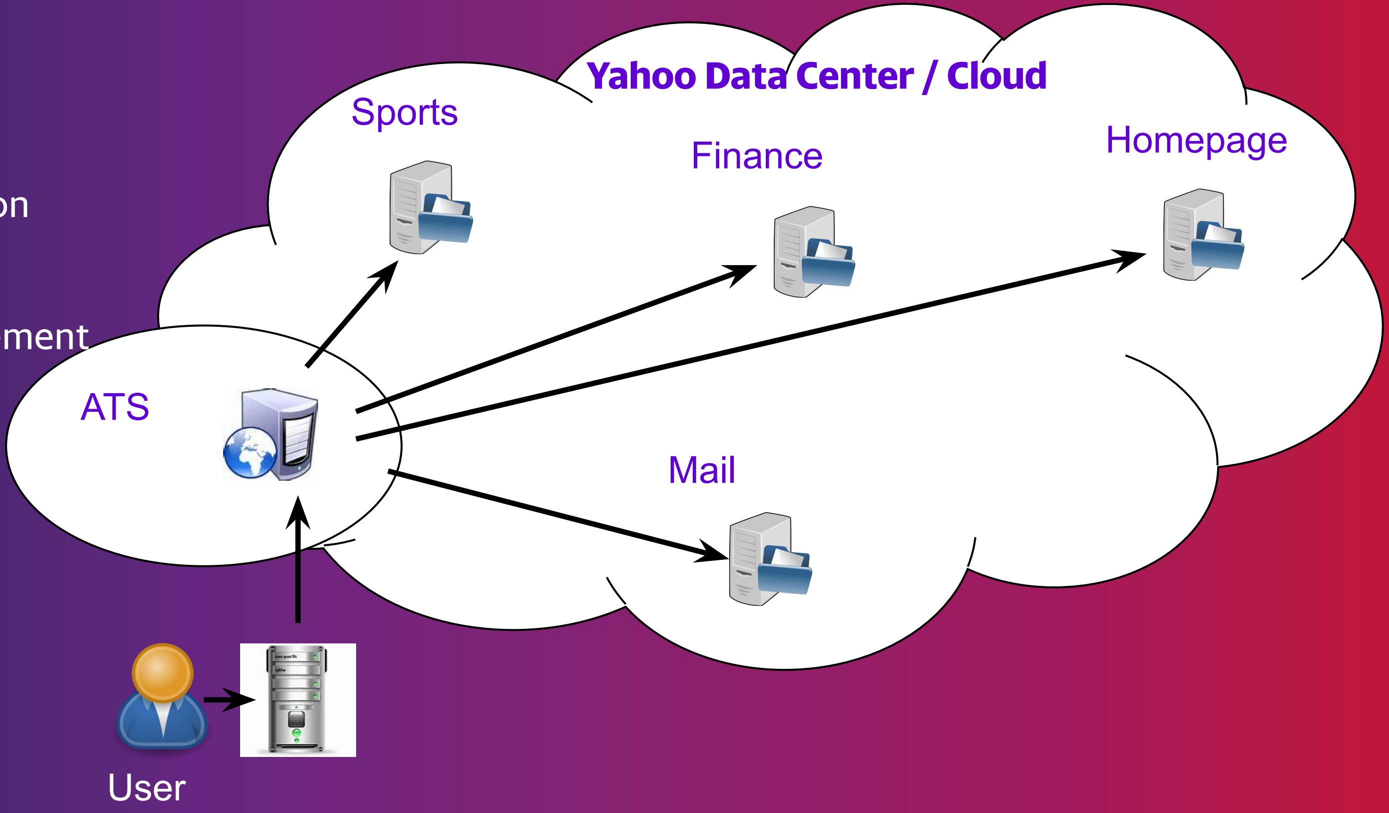
Cookie Management

Routing

Redirect

Bucketing

Caching



Extending ATS

- C++ plugins
  - Allow extension of HTTP/TLS handling for connections with clients and origins
  - Steep Learning curve
- Domain Specific Languages plugin (header_rewrite / txn_box / etc)
  - Invented language, Not turing complete, no unit test framework
  - Hard to expand
- Lua plugin
  - Easier to learn a scripting language
  - LuaJIT FFI allows expansion with shared libraries
  - Popularity?

COMMUNITY OVER CODE

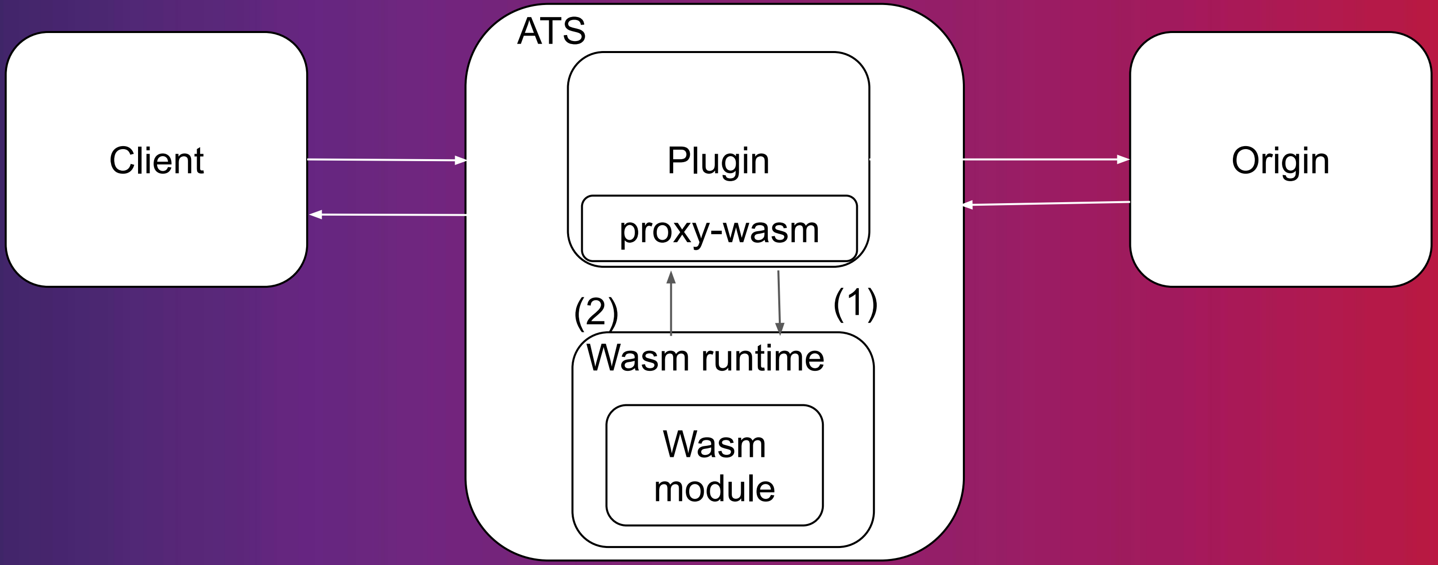
Proxy-Wasm & ATS Wasm Plugin

Proxy-Wasm

- WebAssembly for Proxies
- Specification
    - "WASI for Proxy"
- Library
    - Implement the spec and provide integration with proxy
    - Integrate with different runtime - WAMR, Wasmtime, WasmEdge, V8
    - Existing Implementations - Envoy, MOSN, Nginx, ATS
- SDK
    - Help to compile programs to wasm modules following the spec
    - Official - C++, Rust
    - Third party - AssemblyScript, TinyGo, Zig

High Level Architecture of ATS Wasm Plugin

- Handler functions for proxy to call (1)
- Calling API functions that the proxy provides (2)



Example in Rust (Snippet)

```
impl HttpContext for HttpHeaders {
    fn on_http_request_headers(&mut self, _: usize, _: bool) -> Action {
        for (name, value) in &self.get_http_request_headers() {
            let s3 = format!("In WASM: #{} -> {}: {}",self.context_id, name,
value);
            trace!("{}", s3);
        }
        if let Some(ua) = self.get_http_request_header("User-Agent") {
            if ua != "" {
              trace!("UA is {}", ua);
            }
        }
        match self.get_http_request_header("token") {
            Some(token) if token.parse::<u64>().is_ok() && is_prime(token.parse().unwrap()) => {
                trace!("It is prime!!!");
                Action::Continue
            }
            _ => {
                trace!("It is not prime!!! That's true.");
                self.send_http_response(
                    403,
                    vec![("Powered-By", "proxy-wasm")],
                    Some(b"Access forbidden.\n"),
                );
                Action::Pause
            }
        }
    }
}
```

Real World Example

- WAF
  - Coraza
    - Open Source WAF library in Go
    - Compatible with ModSecurity Ruleset Language
  - Coraza Proxy-Wasm module
    - WASM module to be used with Envoy
    - Compiled with TinyGo SDK
  - It works now with ATS with the Wasm Plugin!
- Other use cases. E.g. AI inference with WASI-nn

Benefits

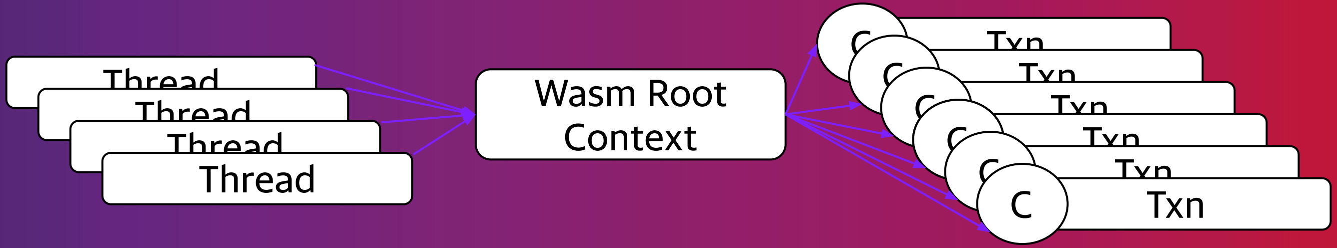
- Support many programming languages
- Standard/specification promote interoperability
- Safety with Sandboxed approach
- Promising future

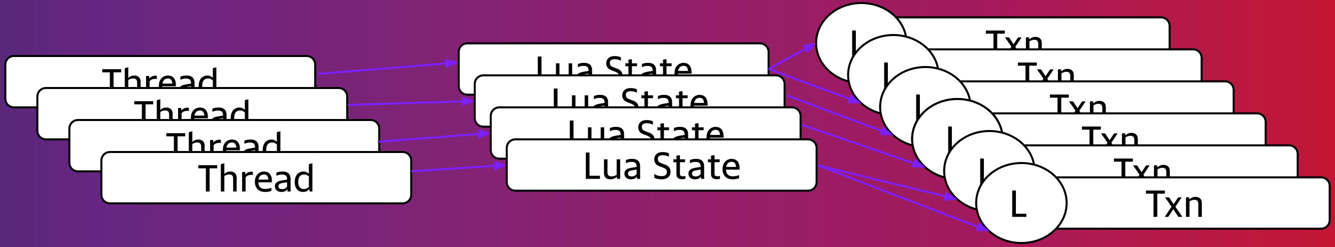
Limitations

- No ATS support in proxy-wasm spec for
  - Getting and setting trailer request and response header
  - Getting and setting data in HTTP/2 meta data frame
  - Support on GRPC lifecycle handler functions
- No proxy-wasm support for ATS Specific features
  - E.g. caching API
  - Can be implemented outside of spec
  - But it will break interoperability

Performance

- Tests between Lua script, DSL script and Wasm module
- Lua script / DSL script < Wasm module (LuaJIT is AWESOME!!!)
- Culprit - Resource Contention inside Wasm plugin





Other Optimizations

- Language Choice
- AOT - ahead of time compilation
- Compiler Flags
- wasm-opt
- Choice of Runtime

COMMUNITY OVER CODE

Wasm Runtimes

Big Decision to Choose

- The field evolves rapidly
- Each with different characteristics
- Change of runtime only possible for simple program
- Major investment involved when tools are used (e.g. profiling / debugging)
    - WAMR/Wasmtime - live debug support through lldb
    - Wasmtime - profiling with perf
- Different WASM proposals supported by different runtime
- Performance
- Trust in Security
    - Choice of implementation language
    - Maturity of processes handling CVE

Runtimes

WAMR

- Bytecode Alliance project
- Written in C
- Interpreter or JIT / LLVM JIT
- Configurable options at compile time
- Low memory footprint

Wasmtime

- Bytecode Alliance project
- Written in Rust
- Based on Cranelift
- High memory footprint

Runtimes

WasmEdge

- Written in C++
- LLVM JIT
- High memory footprint
- Lots of focus on AI Inference use cases

V8

- Not yet supported in ATS Wasm plugin
- Written in C++
- Many dependencies / Complicated to get it to work

COMMUNITY OVER CODE

Summary

ATS Wasm Plugin

- Available now / Another option for extending ATS
- Language supported - C++, Rust, TinyGo, AssemblyScript, Zig
- Runtime supported - WAMR, wasmtime, WasmEdge

To Do

- Performance Improvement
  - Resource contention
  - Test runtimes with different configuration options
- Tooling support
  - Profiling with perf
  - Debugging with lldb
- Explode More Use Cases
  - AI Inference with WASI-nn
- More Runtime Support
  - V8
- Future
  - Component Model
  - WASI HTTP

References

- ATS Plugin development - https://docs.trafficserver.apache.org/en/latest/developer-guide/plugins/index.en.html
- ATS header_rewrite plugin - https://docs.trafficserver.apache.org/en/latest/admin-guide/plugins/header_rewrite.en.html
- ATS Lua plugin - https://docs.trafficserver.apache.org/en/latest/admin-guide/plugins/lua.en.html
- ATS Wasm plugin - https://docs.trafficserver.apache.org/en/latest/admin-guide/plugins/wasm.en.html
- Proxy-wasm - https://github.com/proxy-wasm
- Proxy-wasm spec - https://github.com/proxy-wasm/spec
- Proxy-wasm Library - https://github.com/proxy-wasm/proxy-wasm-cpp-host
- Proxy-wasm C++ SDK - https://github.com/proxy-wasm/proxy-wasm-cpp-sdk
- Proxy-wasm Rust SDK - https://github.com/proxy-wasm/proxy-wasm-rust-sdk
- Rust example - https://github.com/apache/trafficserver/tree/master/plugins/experimental/wasm/examples/rust
- Coraza - https://github.com/corazawaf/coraza
- Coraza Proxy-wasm - https://github.com/corazawaf/coraza-proxy-wasm
- Coraza Proxy-wasm in ATS - https://github.com/apache/trafficserver/tree/master/plugins/experimental/wasm/examples/tinygo
- Wasi-nn - https://github.com/WebAssembly/wasi-nn
- WAMR - https://github.com/bytecodealliance/wasm-micro-runtime
- Wasmtime - https://github.com/bytecodealliance/wasmtime
- WasmEdge - https://github.com/WasmEdge/WasmEdge
- WASI HTTP - https://github.com/WebAssembly/wasi-http

COMMUNITY
OVER CODE

Thank you! Questions?

COMMUNITY OVER CODE

Linkedin Profile

Presentation link