Improving the eBPF Developer Experience With Rust!

Dave Tucker

Alessandro Decina

About us

Dave Tucker

- Principal Software Engineer, Red Hat Office of the CTO
- A Go developer, learning Rust
- Networking & Containers (Docker)

Alessandro decina

- Software Engineer, Deepfence
- Added eBPF support to Rust
- Started Aya

Developer experience

The eBPF Developer Path

1. Get hooked with perf and bpftrace one-liners

2. Identify > 1 line problem that could be solved with eBPF

3. Choose your own adventure:
   a. Use a DSL like bpftrace/systemtap

   b. Use C for the eBPF program and choose a userspace library

   c. Use a single language for both eBPF and userspace

Developer Experience

1. Documentation	Scattered. Evolving quickly
2. Project Bootstrap	Change an example from libbpf-bootstrap
3. Build and Test Loop	Compile. Load. Manually Check Behaviour. Or, try BPF_PROG_TEST_RUN.
4. Debugging	printk

Why RUST?

Why Rust?

Rust is a highly expressive language, comes with a feature rich standard library and can still get as low level as C

Memory safety (userspace) is great. Powerful type system and macros make writing eBPF code easier.

Fantastic dev tools including rustup, cargo, rust-analyzer

AYA

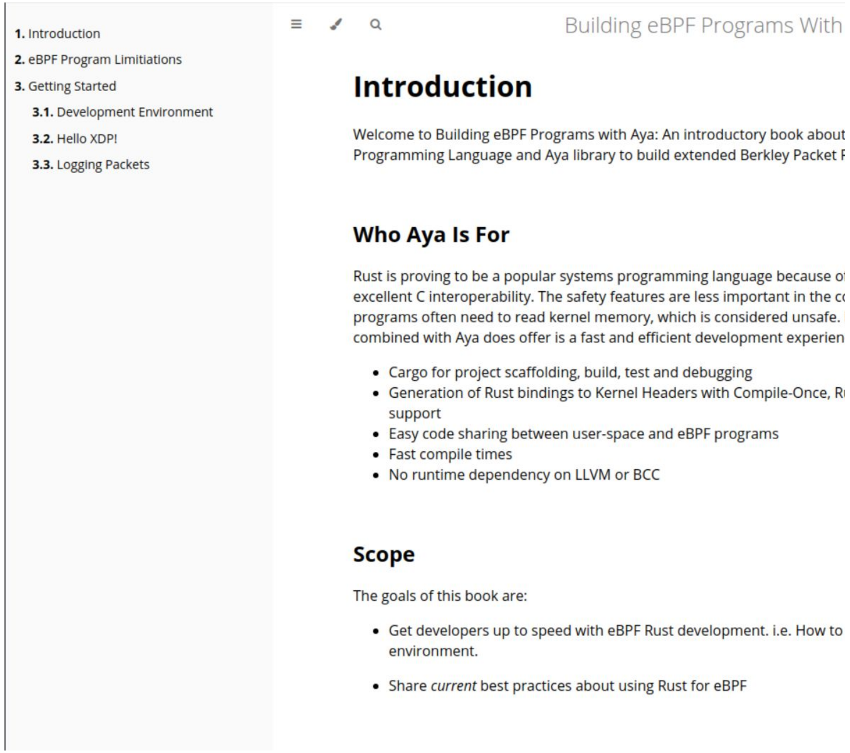
ABOUT AYA

Aya is the first Rust native eBPF library. It provides:

- An userspace eBPF library (like libbpf), completely written in rust
- An high level rust API to write eBPF code - like bpftrace or the bcc DSL - but using plain rust

THE AYA EXPERIENCE

1. DOCUMENTATION



1. Introduction
2. eBPF Program Limitiations
3. Getting Started
   3.1. Development Environment
   3.2. Hello XDP!
   3.3. Logging Packets

Building eBPF Programs With

Introduction

Welcome to Building eBPF Programs with Aya: An introductory book about
Programming Language and Aya library to build extended Berkley Packet F

Who Aya Is For

Rust is proving to be a popular systems programming language because o
excellent C interoperability. The safety features are less important in the c
programs often need to read kernel memory, which is considered unsafe.
combined with Aya does offer is a fast and efficient development experien

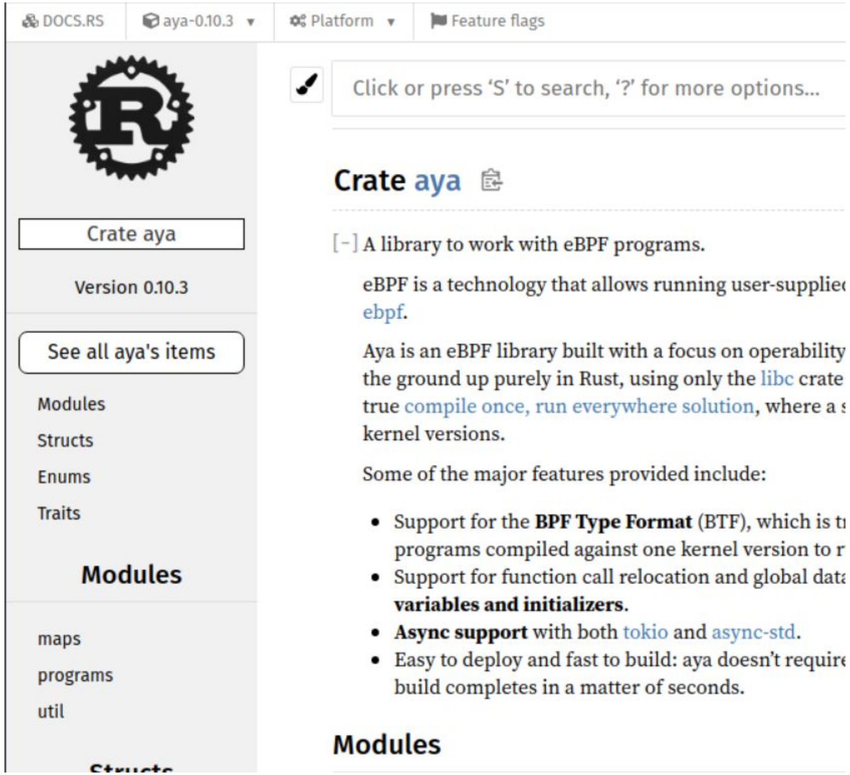
- Cargo for project scaffolding, build, test and debugging
- Generation of Rust bindings to Kernel Headers with Compile-Once, R support
- Easy code sharing between user-space and eBPF programs
- Fast compile times
- No runtime dependency on LLVM or BCC

Scope

The goals of this book are:

- Get developers up to speed with eBPF Rust development. i.e. How to environment.
- Share *current* best practices about using Rust for eBPF

https://aya-rs.github.io/book



DOCS.RS | aya-0.10.3 | Platform | Feature flags

Crate aya

Version 0.10.3

See all aya's items

Modules
Structs
Enums
Traits

Modules

maps
programs
util

Click or press 'S' to search, '?' for more options...

Crate aya

[-] A library to work with eBPF programs.

eBPF is a technology that allows running user-supplie
ebpf.

Aya is an eBPF library built with a focus on operability
the ground up purely in Rust, using only the libc crate
true compile once, run everywhere solution, where a s
kernel versions.

Some of the major features provided include:

- Support for the **BPF Type Format** (BTF), which is t programs compiled against one kernel version to r
- Support for function call relocation and global dat **variables and initializers.**
- **Async support** with both tokio and async-std.
- Easy to deploy and fast to build: aya doesn't requir build completes in a matter of seconds.

Modules

https://docs.rs/aya/

2. BOOTSTRAP

```
$ cargo generate https://github.com/aya-rs/aya-template
🤷   Project Name : lpc2021
🔧   Generating template …
? 🤷   Which type of eBPF program? ›
❯ kprobe
  kretprobe
  uprobe
  uretprobe
  sock_ops
  sk_msg
  xdp
  classifier
  cgroup_skb
  probe
  tracepoint
```

2. BOOTSTRAP

This gives you a workspace with 3 packages:

- lpc2021 (userspace)
- lpc2021-common (code shared between eBPF and userspace)
- lpc2021-ebpf (eBPF code)

2. BOOTSTRAP

A task to generate bindings to kernel types can easily by added:

```
$ cargo xtask codegen
```

This uses aya-gen to create Rust bindings to using the BTF types in /sys/kernel/btf/vmlinux

3. Build and test loop

Build & Run:

```
$ cargo build

$ cargo xtask build-ebpf

$ sudo ./target/debug/myapp --path ./target/bpfel-unknown-none/debug/myapp
```

💡 The second step is required as we need nightly rust to compile eBPF and several unstable cargo features to support having a multi-target workspace. In time, this step will be removed

4. Debugging

Debugging eBPF programs can be hard. Common options include:

- bpf_trace_printk() - slow, hard to follow output with multiple programs
- ad hoc perf events to trace program flow and dump data - works but inconvenient

4. Debugging with aya-log

```
info!(&ctx, "aya-log is a lightweight logging library for eBPF code");

warn!(&ctx, "it sends logs to userspace as perf events");

debug!(&ctx, "it supports string {}", "formatting");

trace!(&ctx, "it integrates nicely with the standard rust log crate");

error!(&ctx, "find it at https://github.com/aya-rs/aya-log");
```

4. Debugging with aya-log

```
07:17:40 [INFO] [src/main.rs:35] aya-log is a lightweight logging library for eBPF code

07:17:40 [WARN] [src/main.rs:36] it sends logs to userspace as perf events

07:17:40 [DEBUG] (4) [src/main.rs:37] it supports formatting

07:17:40 [TRACE] (4) [src/main.rs:38] it integrates nicely with the standard rust log crate

07:17:40 [ERROR] [src/main.rs:39] find it at https://github.com/aya-rs/aya-log
```

ROADMAP

Unit testing

- We plan to add the ability for program contexts and maps to be mocked so code can be tested on the host architecture

- This should speed up the build/test loop significantly

LIBBPF COMPATIBILITY

- Automated tests to ensure libbpf compatibility for implemented program types

- More program types! - LSM and more cgroup hooks are in progress

CRANELIFT

- A code-generator for WebAssembly, written in Rust
- We’re looking to add an eBPF backend, to allow Rust to eBPF compilation

QUESTIONS?

FIND US ON GITHUB

JOIN US ON DISCORD