### eunomia-bpf：eBPF 轻量级开发框架

### eunomia-bpf项目重磅开源！eBPF 轻量级开发框架来了

**郑昱笙**

**浙江大学**

近日，在 2022 云栖大会龙蜥峰会 eBPF & Linux 稳定性专场上，来自eBPF技术探索SIG **Maintainer** 、浙江大学的郑昱笙介绍了《**eunomia-bpf**：eBPF **轻量级开发框架**》，以下为本次演讲内容：



大家好！我是来自浙江大学的郑昱笙，今天为大家介绍下 eunomia-bpf 项目，作为一个为了简化 eBPF 程序的开发、分发、运行而设计的轻量级 eBPF 开发框架的背景和目标；再通过一些简单的实例，展示一下 eunomia-bpf 是如何从云端一行命令下载运行 eBPF 程序、只编写内核态代码即可运行和导出事件，以及和 WebAssembly 的结合等功能，最后简要阐述一下 eunomia-bpf 的原理和设计实现的思路，探讨一下接下来的发展方向。

1. **概要**

eunomia-bpf 起源于今年 2022 年全国大学生操作系统大赛的一个 idea，希望将 eBPF 程序作为服务运行，把 eBPF 程序打包为一个 JSON 对象，通过 HTTP 请求即可动态插拔运行任意一个可重定位的 eBPF 程序，并且可以适应不同内核版本和架构。比赛结束之后，在高校的几位老师和社区中的一些伙伴的帮助和指导下（在这里重点感谢西安邮电大学陈莉君教授及团队和龙蜥社区毛文安老师），逐步把这些想法变成了一个初具雏形的开源项目。目前除了我之外，也主要是陈莉君老师的团队、龙蜥社区的朋友等在一起协作。



当前，eunomia-bpf 想要解决的问题，或者说eBPF程序的开发和分发过程中存的痛点主要有以下2个：

1. 对于新手而言，搭建和开发 eBPF程序的门槛较高，不仅需要同时关注内核态和用户态两方面的交互和信息处理，还需要编写复杂或重复性较高的用户态加载代码。
2. 在不同架构的不同内核版本上无法方便快捷地打包、分发、发布各种eBPF程序。eBPF很多小工具由不同的语言开发，存在不同的接口，无法轻易集成到大型的可观测系统。当前也没有很好的插件方案，对于一个大型的 eBPF 应用，很多时候必须重新编译整个可观测的框架，再重新部署上线，才能更新 eBPF 探针或数据处理模块。另外，如果引入未经审查和测试的第三方的用户态数据处理代码，代码崩溃也可能导致整个程序崩溃。

因此，针对上面两个问题，我们提出了三种解决思路：

1. 针对初学者，只需要编写内核态代码即可自动采样、自动获取内核态导出的数据，编译后即可进行分发、加载和运行，极大地降低了eBPF的学习成本，提高了开发效率。
2. 基于 libbpf一次编译处处运行的特性，将用户态、内核态的编译和运行的完全分离，通过标准 JSON 或 WASM模块的方式进行分发，无需进行重新编译，应用启动占用资源少，时间短，甚至容器启动更短。

* WebAssembly (缩写 Wasm) 是一种基于堆栈虚拟机的二进制格式，Wasm是为了可移植的目标而设计。可作为 C/C+/RUST 等高级语言的编译目标，使客户端和服务器应用程序能够在 Web 上部署。到现在为止，WASM已经发展成为一个轻量级、高性能、跨平台和多语种的软件沙盒环境，被运用于云原生软件组件，可以在非浏览器环境下运行。WASM的设计思路和 eBPF 也有不少相似之处。

第三，只编写内核态代码的时候，使用 JSON 即可完成分发、加载、打包的过程，对于完整的、需要用户态和内核态进行交互的 eBPF 应用或工具，可以在 WASM中编写复杂的用户态处理程序进行控制和处理，并且将编译好的 eBPF 字节码嵌入在 WASM模块中一同分发，在目标机器上动态加载运行；

* 和 WASM生态项结合可以给 eBPF 程序带来许多特性，同时和 eBPF 程序原本的设计思路也不谋而合，比如可移植、隔离性、安全性，它也是一个跨语言、轻量级的运行环境等等；同时也可以借助 WASM的相关工具完成 eBPF 程序的 OCI 镜像的存储和分发，最近 docker 官方也推出了一个基于 WASM的分发和运行工具。

以上三部分就是 eunomia-bpf 的核心特性。接着我们和大家一起来看一些示例。

**二、示例**

eunomia-bpf 并不是一个完整的系统，而是类似于开发库和开发框架，可以很轻松地嵌入coolbpf 这样的大型工具链里，或作为运行时库嵌入到任何需要使用 eBPF 作为插件的的地方。

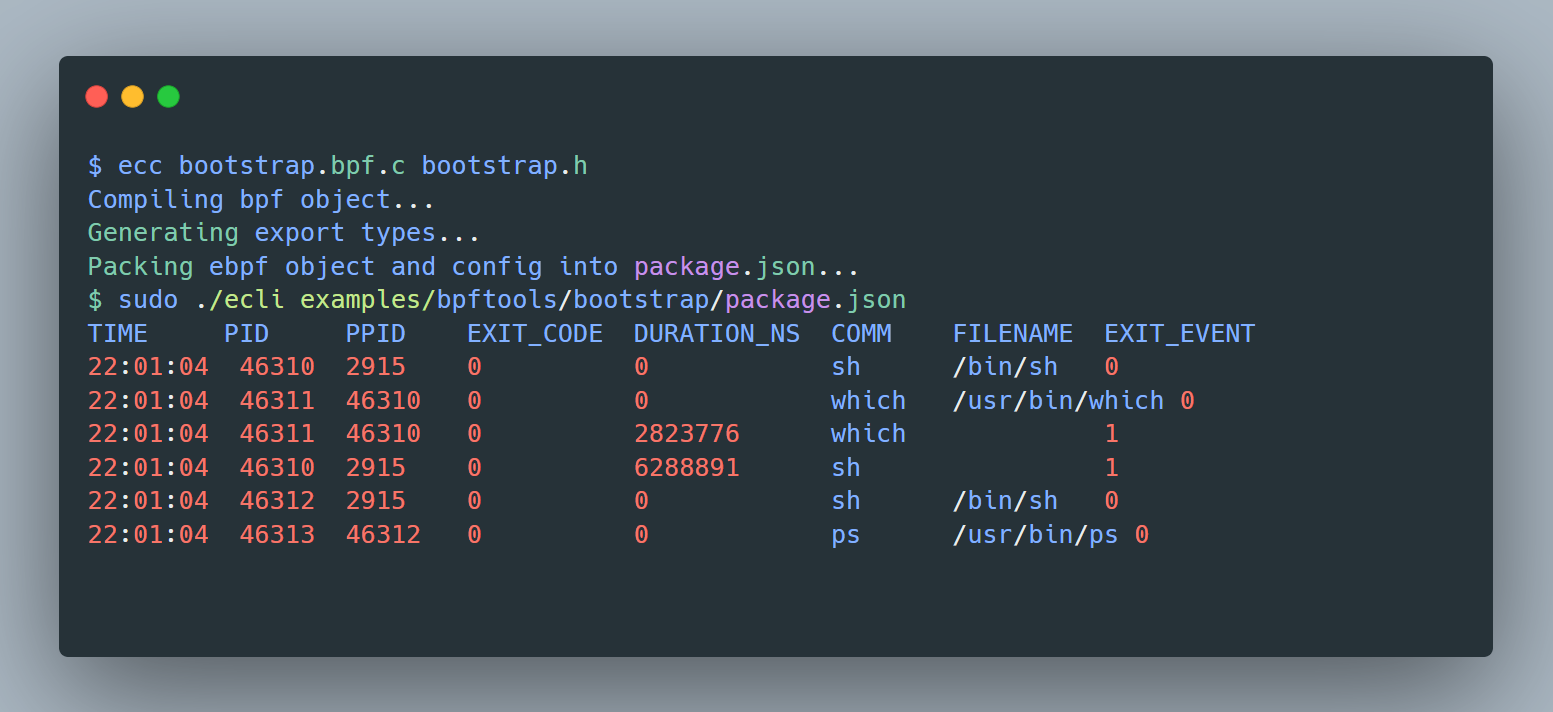


可以通过一行命令从网页端直接下载预编译好的 eBPF程序运行。使用 WebAssembly 模块或JSON配置文件的方式进行分发，部署时无需重新编译，启动速度相比 BCC能提升一到二个数量级。

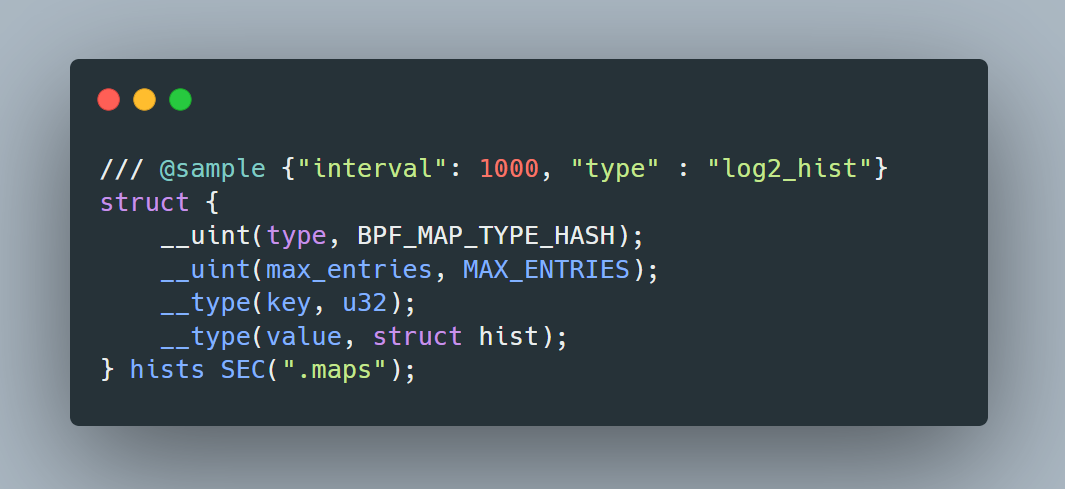
上图中使用URL 启动 eBPF 程序的形式，也可以换成 OCI 镜像或 Docker 镜像，可以存储在 Docker 仓库或 Github Package 中 ，使用方式与Docker 基本一致，只需简单地执行pull、run即可运行，也可以将编译好的程序包 push 下去直接使用。相比于传统的 Docker 镜像，使用 WASM作为轻量级容器的启动速度更快，同时也保留了 eBPF 很重要的特性，可以轻松嵌入到其他程序作为子模块或插件使用，并且和具体内核版本、架构完全无关。



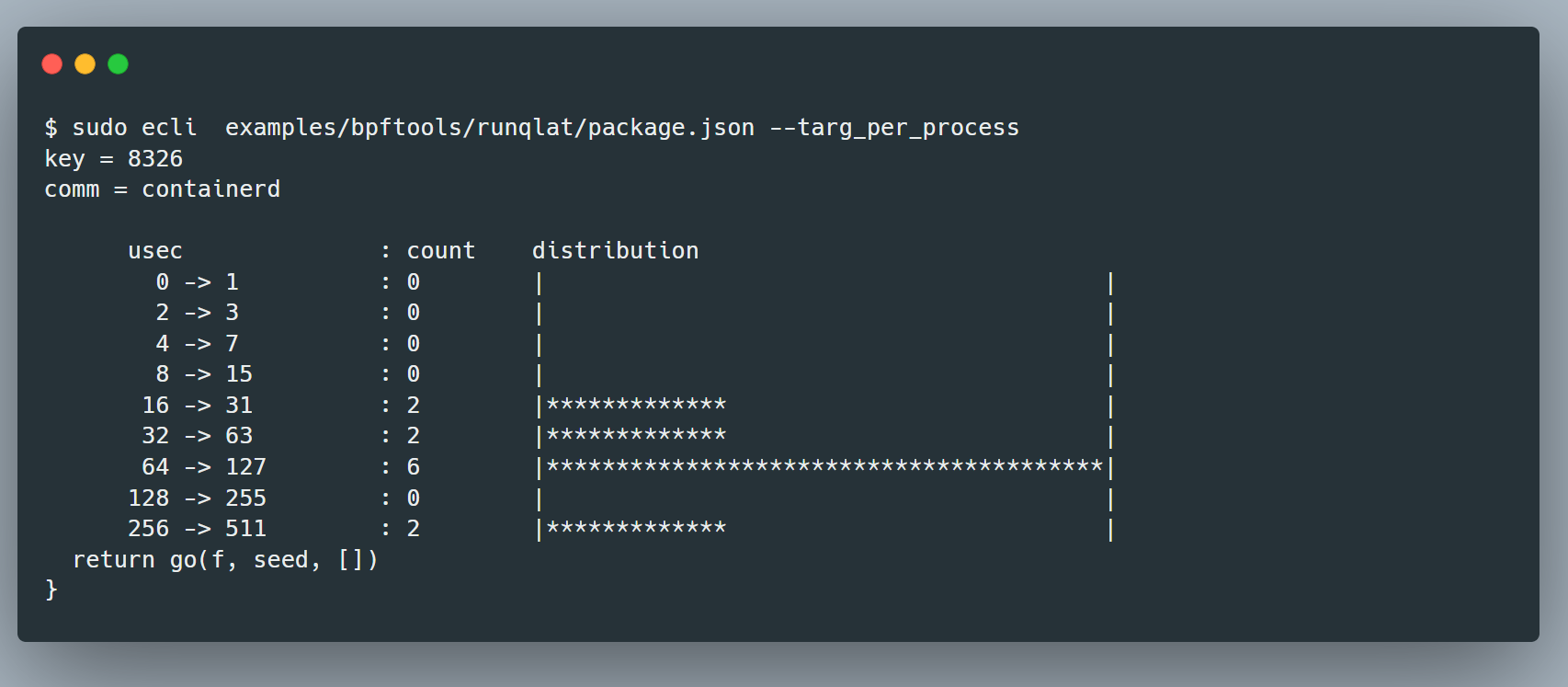
通过eunomia-bpf ，只需编写内核态代码即可正确运行，能够最大程度减少新手的上手障碍，不需要编写 libbpf 的用户态的加载框架，就能够自动导出内核态 perf event 或 ring buffer 事件，例如：



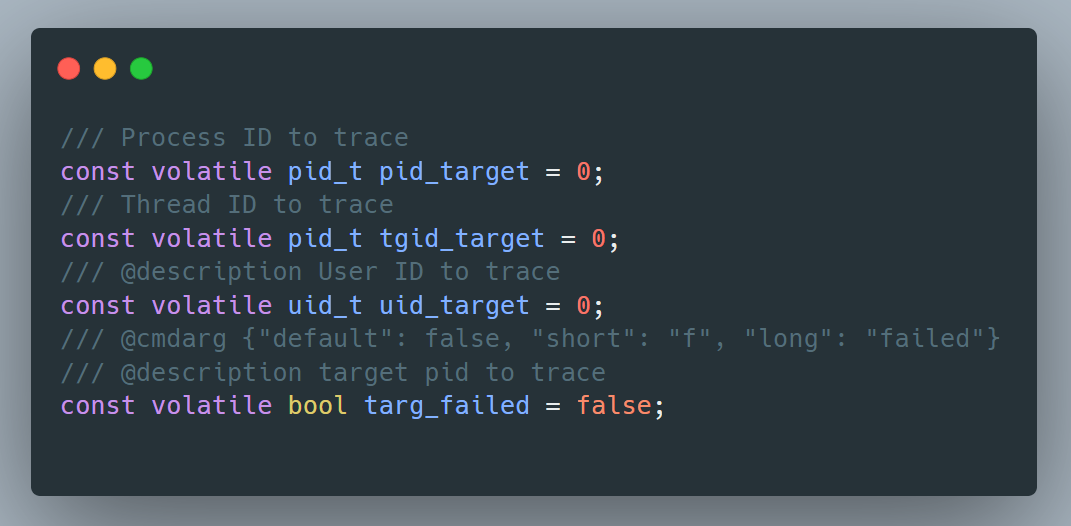
或自动采集 hash map 中的数据，生成直方图等形式，例如如下的代码：



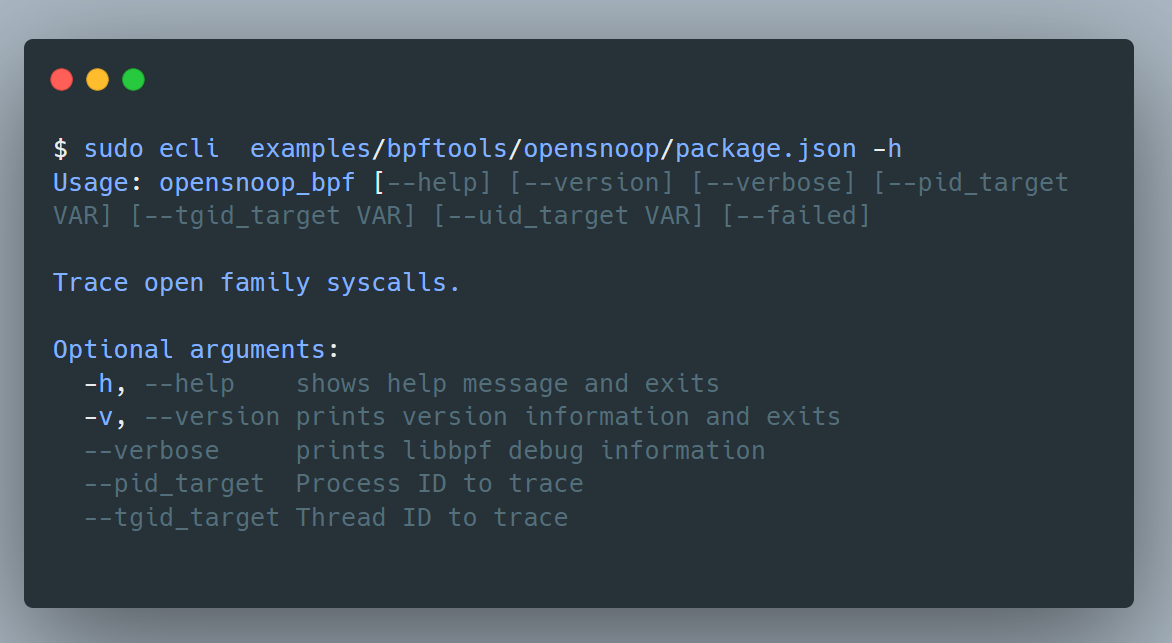
在内核态编写对应的注释信息，就可以自动生成对应的用户态加载代码，并将对应的 hist map 中的信息打印为直方图：



只需在内核态的代码中对应的全局变量位置添加注释，即可自动生成对应工具的命令行参数，例如：



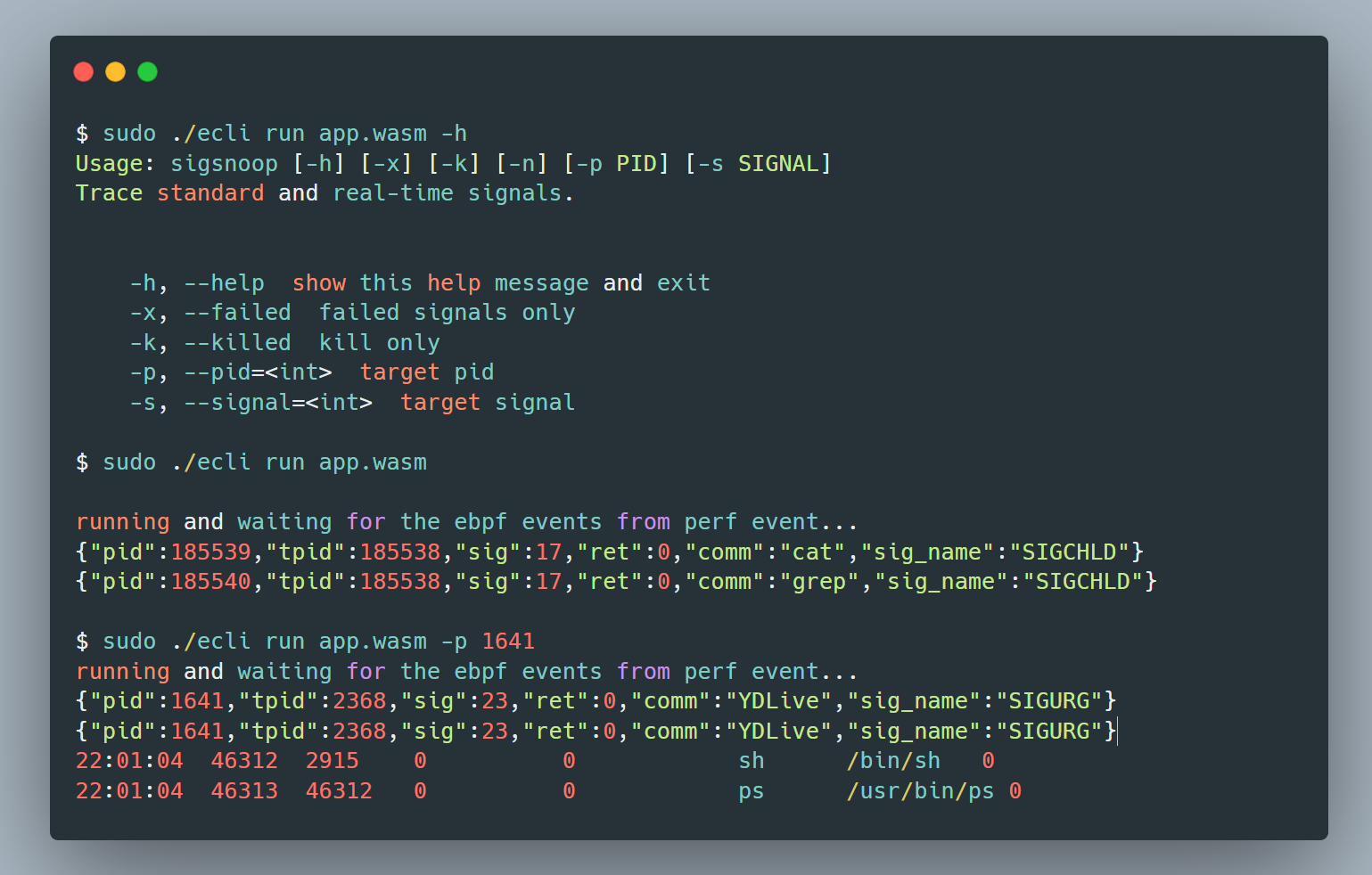
命令行的帮助信息如下：



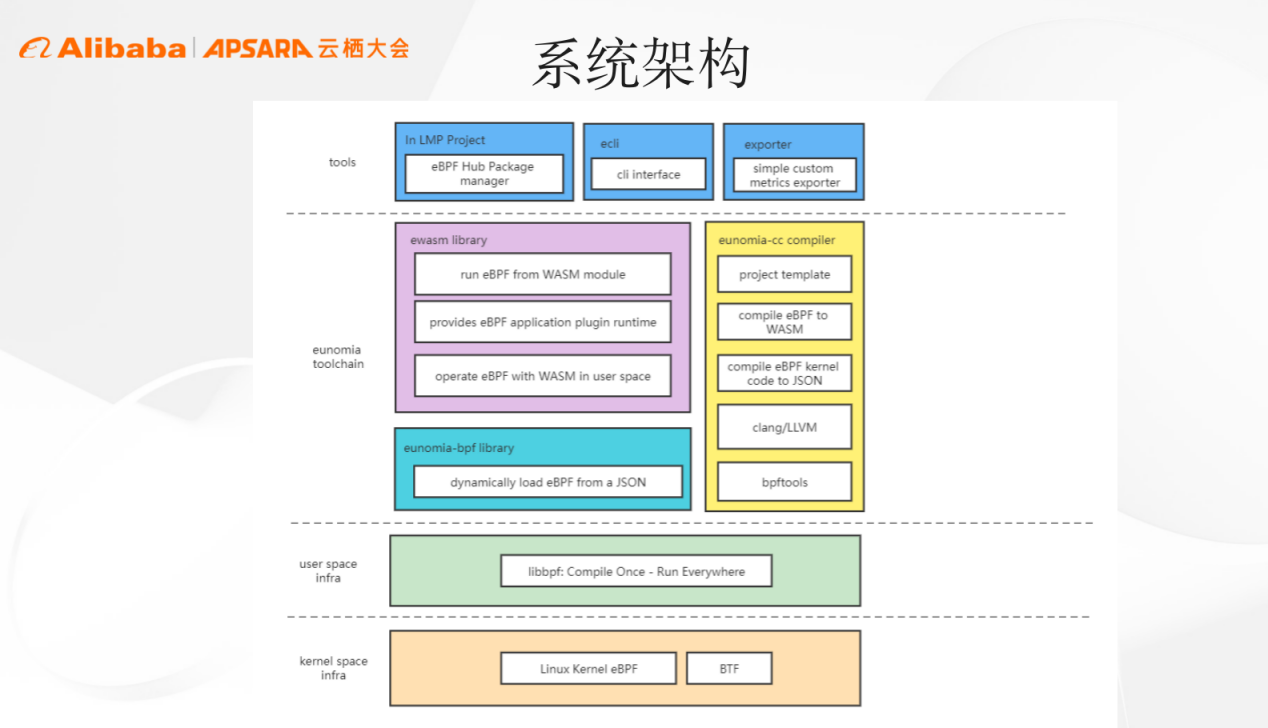
另外，它与和原生 libbpf 完全兼容，可以获取libbpf tools的内核态代码后，无需修改任何代码，可直接打包发布运行。

可以额外添加 tracepoint ，也可以通过注释的形式添加其他内容。使用容器打包编译工具链，无需担心环境配置问题，一行命令生成项目模板、一行命令编译。



一般来说，一个完整的 eBPF 应用程序分为用户空间程序和内核程序两部分，用户空间程序负责加载 BPF 字节码至内核，或负责读取内核回传的统计信息或者事件详情，进行相关的数据处理和控制。  
  
我们可以在 WASM中编写用户态辅助程序，来完成安全、高效的用户态数据处理和控制逻辑，它同样具备 eBPF 的特性，例如安全性（WASM和 eBPF 一样也是个沙盒环境，在用户态运行的时候即使 WASM模块崩溃了，也不会造成宿主程序的异常退出）、可移植性、轻量级、模块化等等，也可以作为插件使用，添加新的数据处理逻辑时，也不需要更改原本的代码。（注意 WASM是可选而不是必须的，对于一些简单的应用而言，编写内核态代码就足够了）  
  
实际上，我们是用 C 语言编写代码，然后打包生成 WASM模块。之后我们可以：  
- 借助 WebAssembly 的相关生态帮助分发、管理 eBPF 程序，例如 docker-wasm  
- 可嵌入大型应用中作为 eBPF 可编程模块或插件使用  
  
这里演示的是一个简单的 WASM模块，它可以获取当前系统的进程间的 signal 信号传递的事件。它可以接受一些命令行参数，并且对上报的信息进行处理。  
  
目前来看，我们已经可以基本上不用进行代码修改，就可以直接把 BCC/libbpf-tools 里面的程序编译为 WASM模块。开发体验来说，也可以做到和使用 C 语言开发 libbpf 的 eBPF 程序完全相同，之后也可以尝试引入其他的语言开发 SDK，例如 go、Rust 、Java等等。  
  
把 WASM和 eBPF 结合起来主要的困难在于，WASM的内存布局和 eBPF 程序并不一样，C 语言的结构体并不能直接映射，所以传递结构体必须要经过序列化操作；同时，WASM对于访问系统资源，例如文件、网络等等，也有不少限制，很多标准库是缺失的，所以我们需要在 WASM模块中进行一些特殊的处理和移植。

**三、系统架构**



架构底层依赖的是内核态和用户态的基础设施，比如 libbpf 库和 kernel ，会提供相关的编译工具链，帮助生成 JSON 或打包成 WASM的模块，工具链本身使用了比如 Clang/LLVM、bpftool等工具。动态加载库可以独立使用，与 WASM无关，可以根据 JSON 信息动态地加载 eBPF程序，可以轻松通过 HTTP 接口，实现 kernel function as your service ，即内核函数及服务的形态。

我们还实现了WASM抽象层，包含 API 规范，比如用于扩展 WASM的虚拟机WSAI 系统占用的访问形式或与 eBPF交互的访问形式。还有基于 WASM定制的 libbpf 库、移植的辅助态程序以及序列化库等，用于在 WASM模块加载基于 libbpf 的 eBPF程序。

运行时库可以轻松进行替换，比如替换成 WSI 的 WASM运行时。除此之外，上层还实现了 LMP、命令行工具、可观测性工具等。

目前，eunomia-bpf 项目已在龙蜥社区开源，欢迎各位开发者体验，也欢迎大家提出建议和反馈，一起来做大做强。

相关链接：

eunomia-bpf 项目地址链接：<https://github.com/eunomia-bpf/eunomia-bpf>

libbpf-wasm 库地址：https://github.com/eunomia-bpf/libbpf-wasm

eBPF技术探索SIG地址：https://openanolis.cn/sig/ebpf-research

**关于龙蜥峰会eBPF & Linux稳定性专场课件获取方式：**

【PPT 课件获取】：关注微信公众号（OpenAnolis），回复**“龙蜥课件”** 即可获取。有任何疑问请随时咨询龙蜥助手—小龙（微信：openanolis\_assis)。

【视频回放】：视频回访已上传至龙蜥官网（可阅读原文直达）：https://openanolis.cn/video 查看。