第三届 eBPF开发者大会

www.ebpftravel.com

eBPF 发展趋势分析

王聪 <xiyou.wangcong@gmail.com>

2025年4月19日

中国·西安

eBPF 发展历程回顾

年份	内核版本	主要新功能
2014	3.18	引入 eBPF 虚拟机、bpf() 系统调用、程序验证器、首批辅助函数（如 map_lookup、trace_printk 等）
2015	4.1	支持内核跟踪：添加 kprobe、tracepoint、perf_event 等程序类型
2016	4.4 / 4.8	引入 XDP（eXpress Data Path），实现高性能包处理
2017	4.10 / 4.13 / 4.14	增加对 cgroup 的支持（如 ingress/egress）、sockops（4.13）、sockmap（4.14）
2018	4.14 / 4.18	支持 sk_msg 用于七层协议解析；引入 BTF（BPF 类型格式）用于类型描述与验证
2019	5.0 / 5.2	支持 LSM（安全模块）钩子；增强 bpf_trace 与原始 tracepoint 支持；扩展辅助函数
2020	5.7 / 5.8	正式引入 LSM 程序类型（BPF_PROG_TYPE_LSM）；支持可睡眠程序（BPF_F_SLEEPABLE）
2021	5.10	支持 CO-RE（编译一次，多处运行）；支持 .rodata、.bss 和全局变量
2022	5.16 / 5.17	支持调用内核函数（kfunc），引入动态对象引用与 bpf_obj_new() 分配器
2023	6.0 – 6.3	引入 kptr 类型，支持在 map 中存储带引用计数的内核指针
2024	6.6 / 6.7	扩展 BPF 运行时类型系统，增强与 Rust 的集成，改进 XDP 多缓冲区支持

eBPF 程序类型增长

内核版本	加载类型	新引入的类型
3.18	1	BPF_PROG_TYPE_SOCKET_FILTER
4.9	6	BPF_PROG_TYPE_KPROBE, BPF_PROG_TYPE_TRACEPOINT, BPF_PROG_TYPE_PERF_EVENT, BPF_PROG_TYPE_SCHED_CLS, BPF_PROG_TYPE_SCHED_ACT
4.19	13	BPF_PROG_TYPE_XDP, BPF_PROG_TYPE_CGROUP_SKB, BPF_PROG_TYPE_CGROUP_SOCK, BPF_PROG_TYPE_CGROUP_DEVICE, BPF_PROG_TYPE_CGROUP_SOCK_ADDR, BPF_PROG_TYPE_CGROUP_SOCKOPT, BPF_PROG_TYPE_CGROUP_SYSCTL
5.10	20	BPF_PROG_TYPE_LSM, BPF_PROG_TYPE_SK_LOOKUP, BPF_PROG_TYPE_SK_REUSEPORT, BPF_PROG_TYPE_FLOW_DISSECTOR, BPF_PROG_TYPE_NETFILTER, BPF_PROG_TYPE_RAW_TRACEPOINT, BPF_PROG_TYPE_RAW_TRACEPOINT_WRITABLE
6.1	25	BPF_PROG_TYPE_TRACING, BPF_PROG_TYPE_EXT, BPF_PROG_TYPE_STRUCT_OPS, BPF_PROG_TYPE_LIRC_MODE2, BPF_PROG_TYPE_SYSCALL
6.12	26	BPF_PROG_TYPE_STRUCT_OPS (enhanced with sched_ext_ops)

辅助函数增长

内核版本	helper 数量	kfunc 数量	备注
3.18	6	0	初始引入 eBPF，仅支持基础功能（map 操作、时间函数等）
4.9	30+	0	增加调度器、网络分类、跟踪相关的辅助函数
4.19	60+	0	引入 BTF，辅助函数类型扩展到 tracing、cgroup 等
5.10	100+	0	LSM 支持上线，新增多个类型和 tracing 相关 helper
6.1	150+	~70	初次引入 kfunc，主要支持 struct_ops、RCU 相关函数
6.12	150+	~270	kfunc 使用范围扩大，支持 bpf_obj_new 等动态对象操作

Linux 内核社区开发

内核版本	BPF总行数	验证器行数	验证器占比
v3.18	~5,000	~2,000	~40%
v4.9	~18,000	~5,000	~28%
v4.19	~35,000	~8,000	~23%
v5.10	~52,000	~12,000	~23%
v6.1	~68,000	~17,000	~25%
v6.12	~80,000	~21,000	~26%

行业会议eBPF专题

会议	eBPF专题数量	主题
eBPF Summit	30+	eBPF 应用
FOSDEM	5+	内核, 网络, 安全
KubeCon	15+	容器网络, 安全策略, 可观测
LSF/MM/BPF	30+	验证器, 内核集成
Linux Plumbers	24	子系统变更

学术研究趋势

年份	论文数量	累计引用次数
2016	3-4	20-40
2017	4-5	30-60
2018	5-7	50-100
2019	6-8	80-150
2020	8-10	120-200
2021	10-12	150-250
2022	12-15	200-300
2023	15-18	250-350
2024	18-22	300-400

eBPF 2025年现状

- 驱动程序 LIRC 和 HID 已经使用 eBPF
- eBPF 已经可以实现功能丰富的 CPU 调度器
- eBPF 也已经可以编写网络队列（Qdisc）
- 支持 arena，可以实现更高效的内存分配
- 支持 tokens，细粒度控制 eBPF 权限
- 目前还没有支持程序签名，社区还在讨论中

总体趋势

- 安全与可观测性融合
- 云原生集成
- 性能优化
- 跨平台支持
- eBPF 编程模型演进

安全与可观测性融合

- 安全工具与可观测性的融合日益突出
- eBPF安全检查:
  - 系统调用
  - 网络流量
  - 应用行为
- 无需修改内核或应用程序
- 实现复杂的运行时威胁检测
- 支持实时策略执行

云原生集成

- Kubernetes和容器平台将eBPF作为基础构建模块
- 主要实现:
  - 网络策略
  - 服务网格实现
  - 容器安全工具
- 与传统方法相比提供更好的性能
- 提供更精细的控制

性能优化

- eBPF程序的开销持续降低（比如raw tracepoint）
- bpftool + perf 工具支持
- 改进领域：
  - JIT 编译器
  - 验证器
  - 内核优化
- 对系统性能的影响最小
- 支持更复杂的编程模型

跨平台支持

- 最初仅限于 Linux 操作系统
- 向其他操作系统扩展的趋势增强
  - 「eBPF for Windows」等项目已经成熟
- 实现跨环境的一致可观测性
- 在异构系统上提供统一的解决方案

编程模型演进

- 得益于 libbpf 开发者体验显著提升
- 动态内存分配、管理、引用计数
- 高级数据结构的支持（链表、红黑树、 arena）
- 逐步增强的 struct_ops
- 逐渐接近内核模块的编程能力

未来方向：存储子系统扩展

- 优化存储性能，用 eBPF 直接操作 bio
- 应用程序定义的 I/O
- 用 eBPF 实现 block I/O 调度策略
- 与 io_uring 深度集成，在内核空间实现更多应用逻辑

未来方向：内存管理扩展

- 用户定制的内存缓存策略，取代 LRU
- 用户定制的 OOM 策略，取代 oom score
- 高度定制化的缺页中断，取代 userfaultfd
- 定制 NUMA 调度策略
- 集成 eBPF 到 DAMON 策略中

未来方向：AI/ML集成

- 自适应安全：ML模型从eBPF跟踪学习"正常"系统行为并实时标记异常
- 智能自动扩缩：结合eBPF资源分析与强化学习优化自动扩缩策略
- 预测性可观测性：关联eBPF数据与性能指标
- 内核机器学习：在eBPF代码中直接实现专门的ML逻辑

未来方向：异构硬件+标准化

- IETF 标准制定
- API 稳定性：跨平台标准化接口
- 硬件卸载：针对 SmartNIC 和可计算存储设备的标准化
- 一致性测试：验证程序行为的一致性
- 高性能计算：GPU 性能分析

未来方向：新的编程模型

- 建立统一的 eBPF 数据流水线，不只是用 map
- 可重入的 eBPF 程序，coroutine
- 集成机器学习训练/推理能力（浮点运算支持？）
- 应用程序快路径下沉到内核中（配合 io_uring）

总结

- eBPF 技术依然在继续快速发展
- eBPF 编程能力逐步增强，生态系统不断扩大
- 未来几年依然是值得关注的 Linux 核心技术
- 对现代云计算基础设施的影响日益增长

谢谢!