CHINA LINUX KERNEL
中国Linux内核开发者大会

華中科技大學
网络空间安全学院
School of Cyber Science and Engineering, HUST

第19届中国
Linux内核开发者大会

赞助单位

華中科技大學 | HUAWEI | 阿里云 | FUJITSU | 腾讯云
OPPO | 火山引擎 | vivo | 龙芯中科 LOONGSON TECHNOLOGY | 蚂蚁集团 ANT GROUP

支持单位

清華大學 Tsinghua University | 迪捷软件 DIGIPROTO | openEuler | OpenAnolis 龙蜥社区 | OpenCloudOS

支持社区&媒体

CSDN | 云巅论剑 | sf 思否 | 阅码场 yomocode | InfoQ 极客传媒
51CTO | 开源江湖

2024年10月　湖北 • 武汉

RISC-V eBPF 现状与未来

蒲乐辉

Pu Lehui <pulehui@huawei.com>

目录

01 RISC-V eBPF 背景

02 RISC-V eBPF 现状

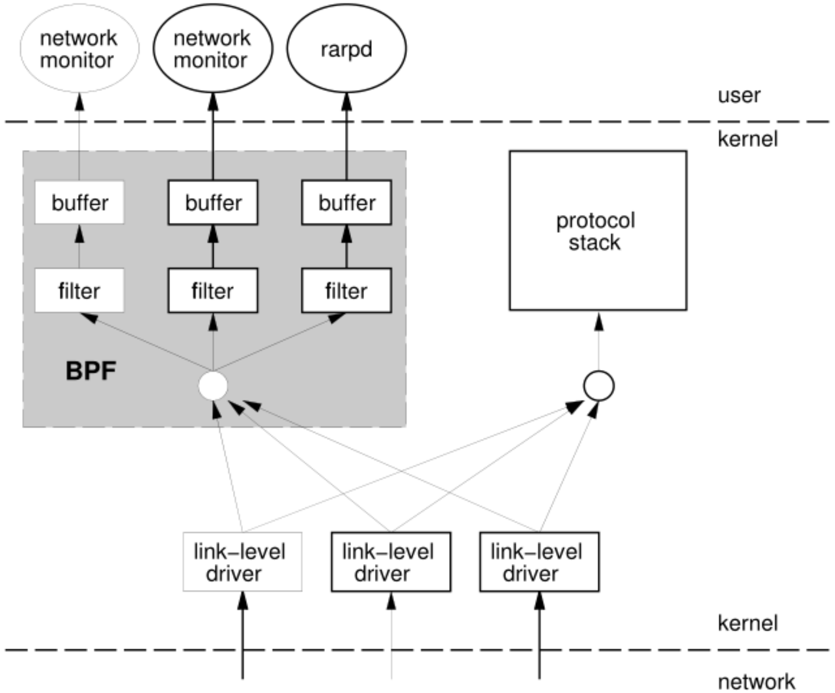
03 RISC-V eBPF 规划

01 RISC-V eBPF 背景

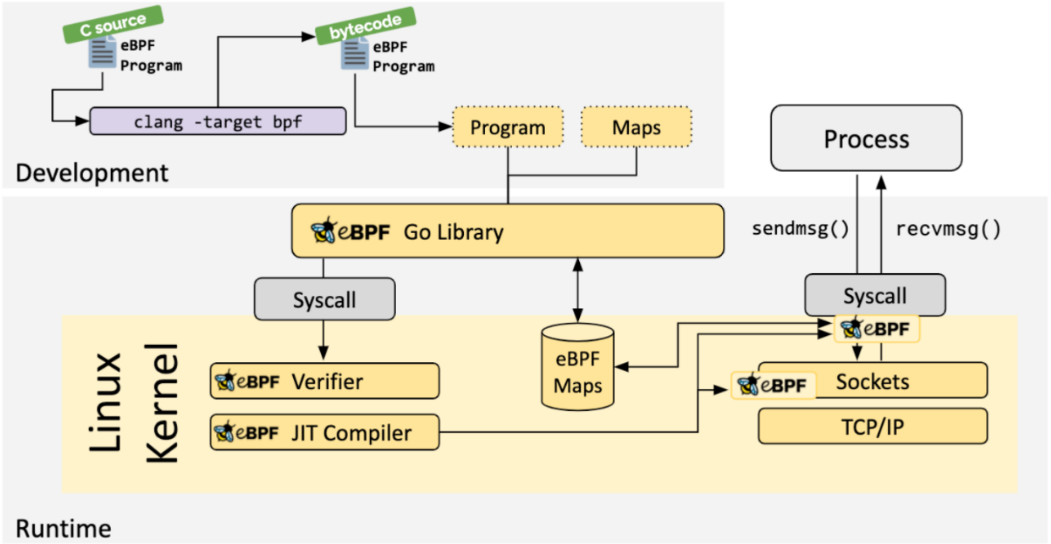
eBPF 介绍

BPF是一种基于虚拟机模型的网络报文过滤器，通过将用户态定义的过滤规则编译成字节码注入内核，内核态虚拟机执行这段字节码完成报文过滤。

eBPF衍生自网络报文过滤器BPF。通过扩充寄存器数量、位宽，引入verifier和MAP机制，将BPF扩展为一个**通用**的内核态虚拟机。



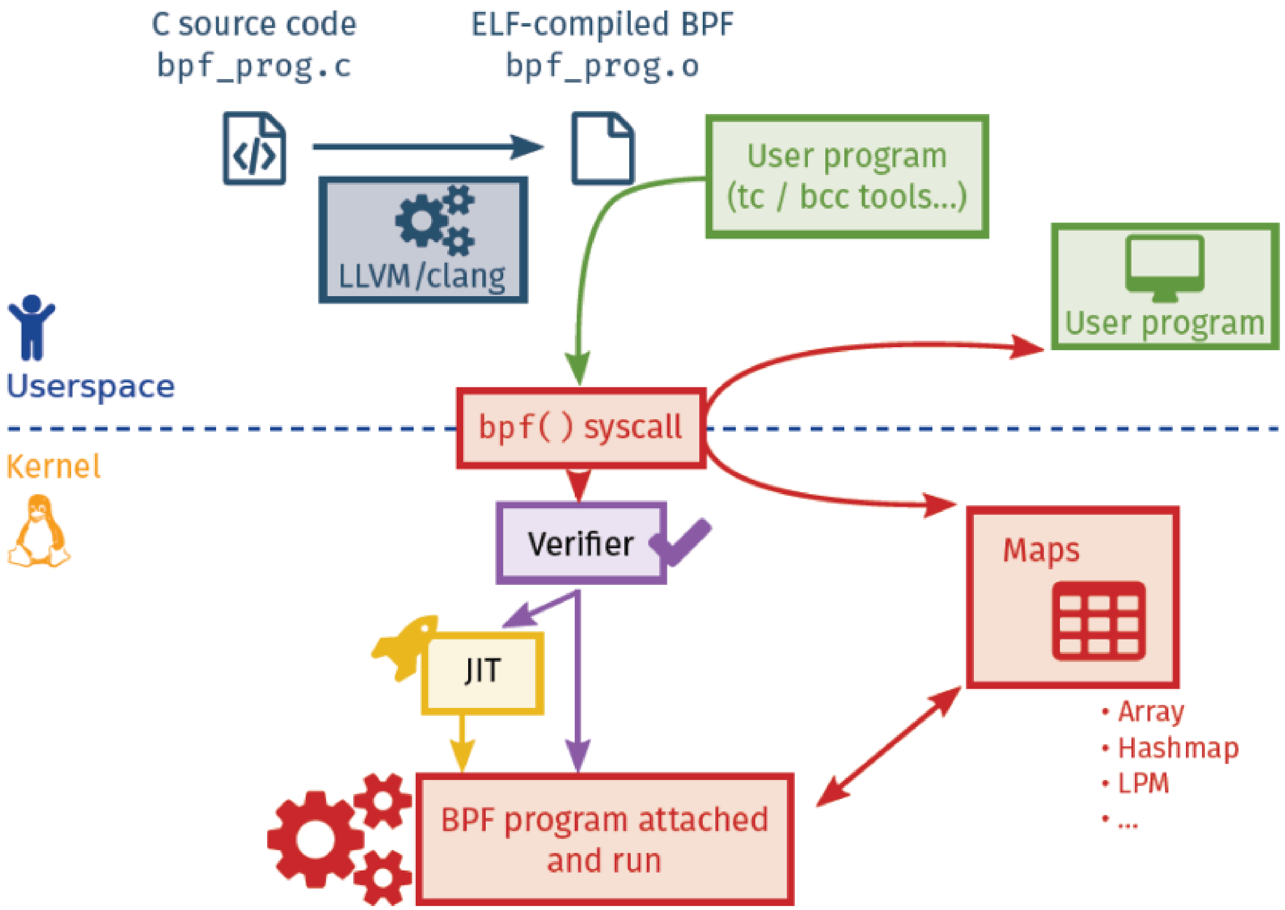
Berkeley Packet Filter



Extended BPF

eBPF 框架

- 加载（load）
  - libbpf解析
    - 整合
    - 重定位
    - CO-RE
    - ……
  - verifier校验
    - 访问校验
    - 分支遍历
    - fixup改写
    - ……
  - JIT编译
    - BPF字节码 -> 机器码
- 挂载（attach）
  - tracing
  - xdp
  - ……
- 触发（trigger）
  - event
  - ……

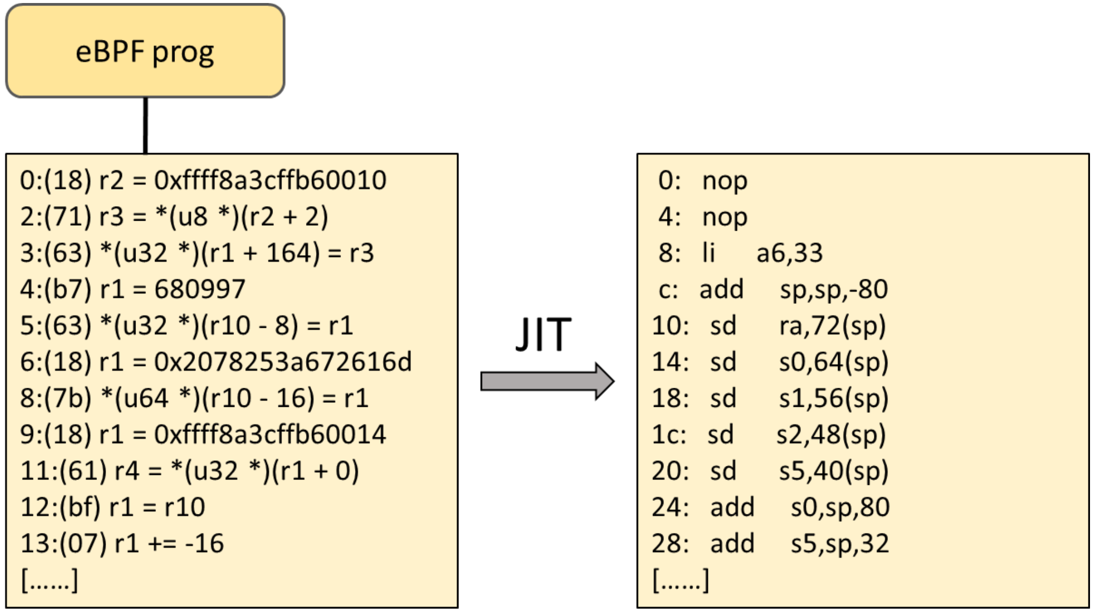


RISC-V eBPF 实现

- JIT：即时编译器，将BPF字节码翻译成RV机器码

```
for (i = 0; i < NR_JIT_ITERATIONS; i++) { // 多轮JIT，优化指令数
    emit prologue: 函数堆栈初始化，保存寄存器等
    emit body: 将一条条BPF指令转换成RV机器码
    emit epilogue: 函数堆栈清理，恢复寄存器等
}
```

- BPF trampoline
- 其他架构实现
  - bpf_arch_text_poke
  - bpf_arch_text_copy
  - bpf_jit_alloc_exec
  - ……



```
0:(18) r2 = 0xffff8a3cffb60010
2:(71) r3 = *(u8 *)(r2 + 2)
3:(63) *(u32 *)(r1 + 164) = r3
4:(b7) r1 = 680997
5:(63) *(u32 *)(r10 - 8) = r1
6:(18) r1 = 0x2078253a672616d
8:(7b) *(u64 *)(r10 - 16) = r1
9:(18) r1 = 0xffff8a3cffb60014
11:(61) r4 = *(u32 *)(r1 + 0)
12:(bf) r1 = r10
13:(07) r1 += -16
[......]
```

```
 0:  nop
 4:  nop
 8:  li      a6,33
 c:  add     sp,sp,-80
10:  sd      ra,72(sp)
14:  sd      s0,64(sp)
18:  sd      s1,56(sp)
1c:  sd      s2,48(sp)
20:  sd      s5,40(sp)
24:  add     s0,sp,80
28:  add     s5,sp,32
[......]
```

02 RISC-V eBPF 现状

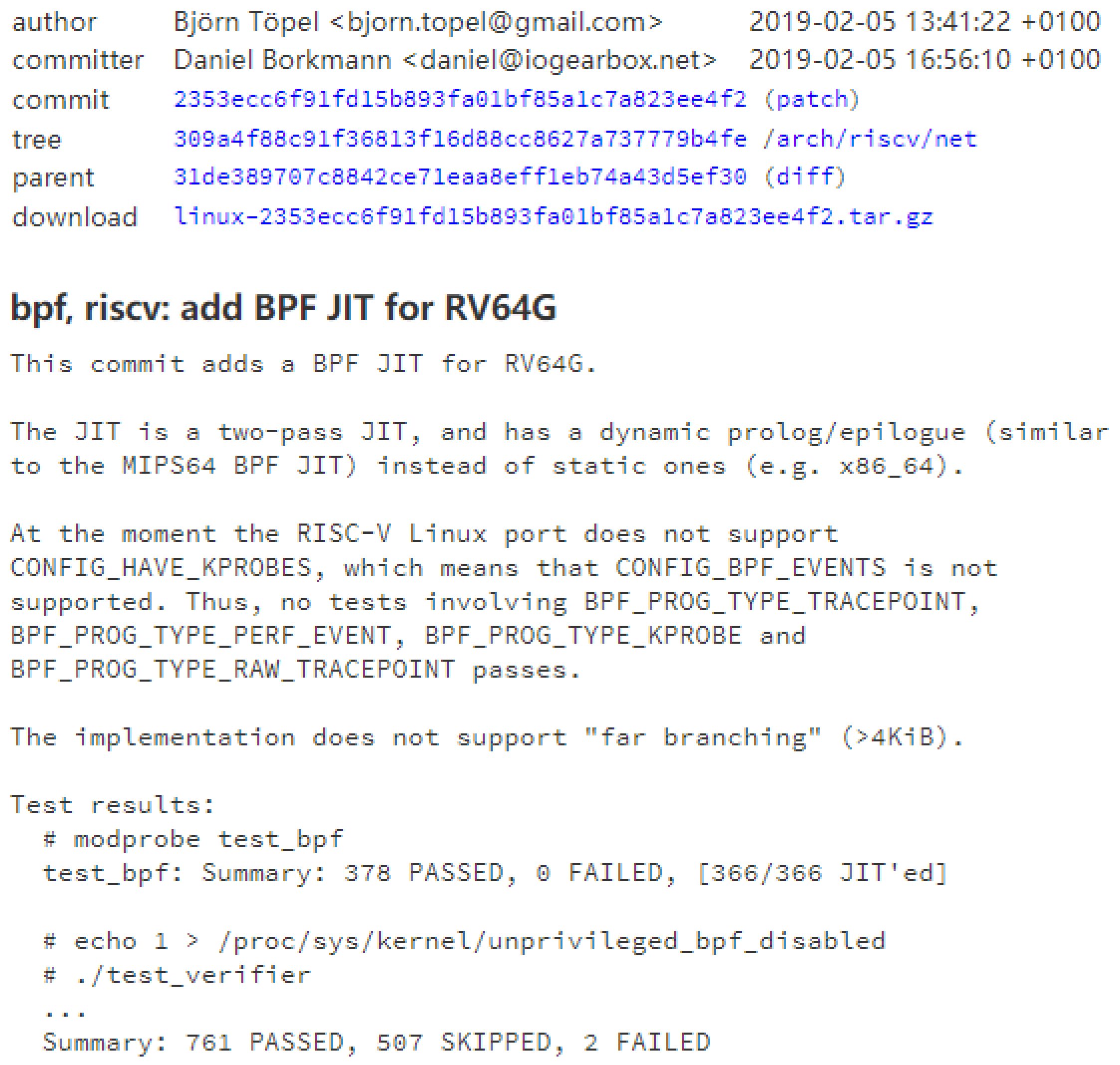
RV64 BPF JIT支持

- 基本框架
  - 寄存器映射：BPF寄存器<->RV寄存器

BPF寄存器	RISC-V寄存器
R0: return value	A5
R1-R5: arguments	A0-A4
R6-R9: callee saved	S1-S4
R10: FP	S5

  - emit prologue
  - emit body
  - emit epilogue
  - BPF基本指令

- 基本指令
  - 运算：ADD/SUB/AND/OR等
  - 跳转：JMP/CALL/TAILCALL等
  - 内存访问：LDX/STR等



author	Björn Töpel <bjorn.topel@gmail.com>	2019-02-05 13:41:22 +0100
committer	Daniel Borkmann <daniel@iogearbox.net>	2019-02-05 16:56:10 +0100
commit	2353ecc6f91fd15b893fa01bf85a1c7a823ee4f2 (patch)	
tree	309a4f88c91f36813f16d88cc8627a737779b4fe /arch/riscv/net	
parent	31de389707c8842ce71eaa8eff1eb74a43d5ef30 (diff)	
download	linux-2353ecc6f91fd15b893fa01bf85a1c7a823ee4f2.tar.gz	

bpf, riscv: add BPF JIT for RV64G

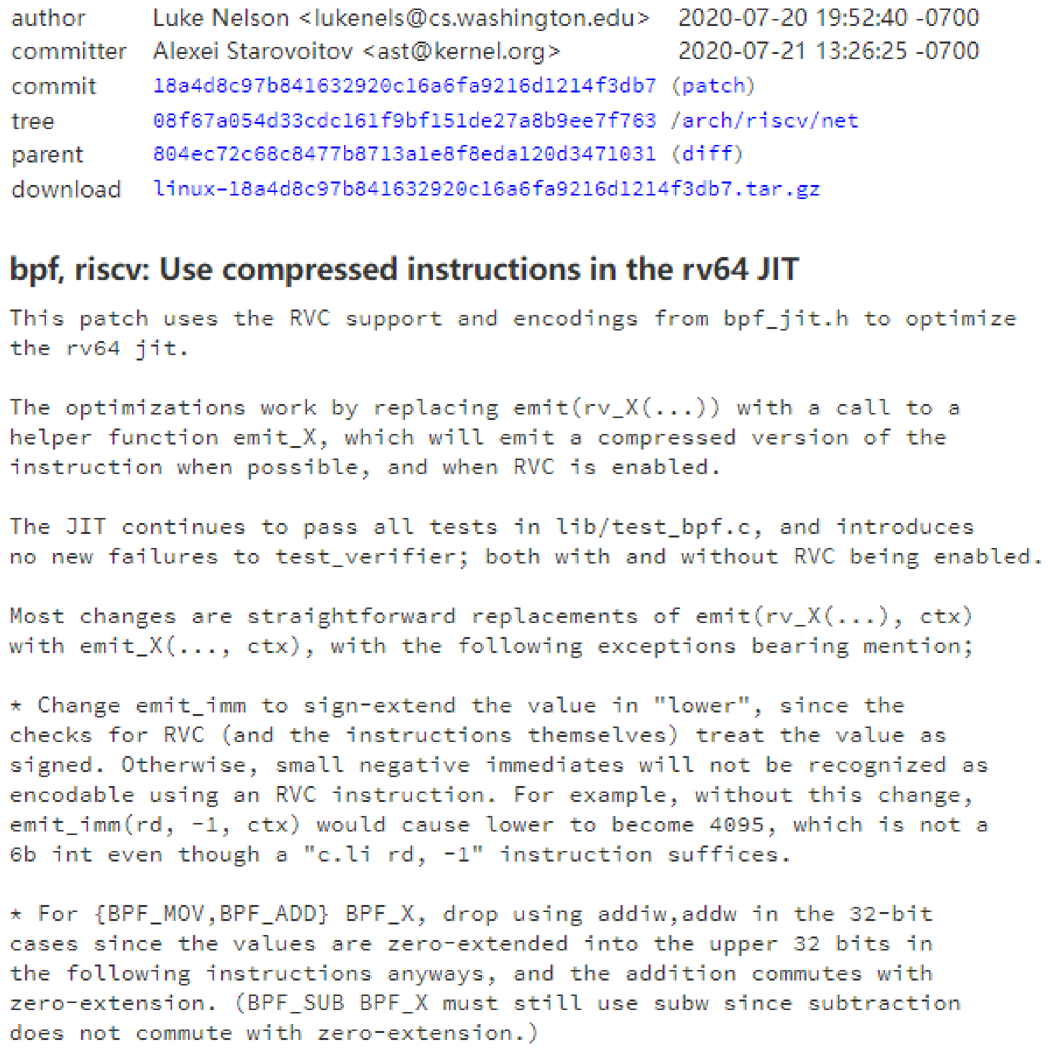
```
This commit adds a BPF JIT for RV64G.

The JIT is a two-pass JIT, and has a dynamic prolog/epilogue (similar
to the MIPS64 BPF JIT) instead of static ones (e.g. x86_64).

At the moment the RISC-V Linux port does not support
CONFIG_HAVE_KPROBES, which means that CONFIG_BPF_EVENTS is not
supported. Thus, no tests involving BPF_PROG_TYPE_TRACEPOINT,
BPF_PROG_TYPE_PERF_EVENT, BPF_PROG_TYPE_KPROBE and
BPF_PROG_TYPE_RAW_TRACEPOINT passes.

The implementation does not support "far branching" (>4KiB).

Test results:
  # modprobe test_bpf
  test_bpf: Summary: 378 PASSED, 0 FAILED, [366/366 JIT'ed]

  # echo 1 > /proc/sys/kernel/unprivileged_bpf_disabled
  # ./test_verifier
  ...
  Summary: 761 PASSED, 507 SKIPPED, 2 FAILED
```

RV64 BPF 压缩扩展支持

CHINA LINUX KERNEL
中国Linux内核开发者大会

- RISC-V指令集
  - 模块化
  - 通用指令集：I, M, A, F, D
  - 扩展指令集：C, Zbb, Sscofpmf

- C Extension/压缩指令扩展
  - 指令宽度：32-bit -> 16-bit
  - 减少镜像体积
  - 增加代码密度，提升性能



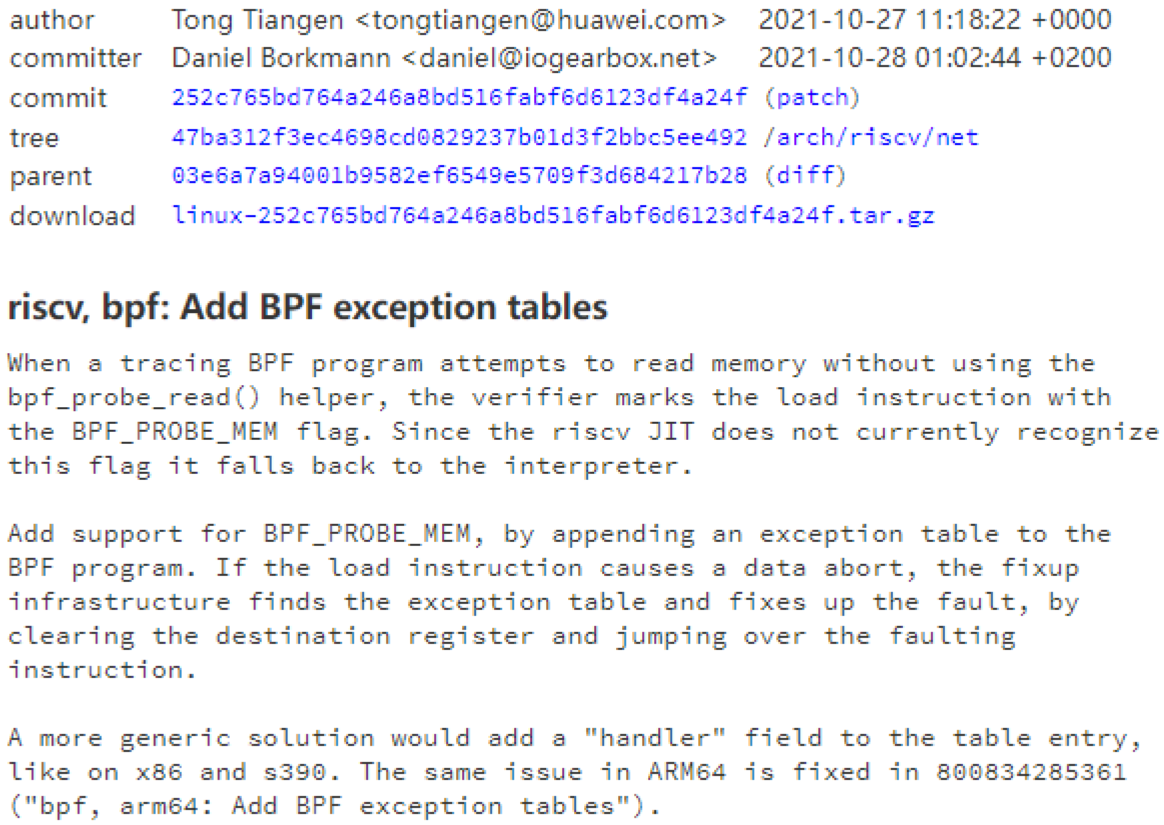
author	Luke Nelson <lukenels@cs.washington.edu>	2020-07-20 19:52:40 -0700
committer	Alexei Starovoitov <ast@kernel.org>	2020-07-21 13:26:25 -0700
commit	18a4d8c97b841632920c16a6fa9216d1214f3db7 (patch)	
tree	08f67a054d33cdc161f9bf151de27a8b9ee7f763 /arch/riscv/net	
parent	804ec72c68c8477b8713a1e8f8eda120d3471031 (diff)	
download	linux-18a4d8c97b841632920c16a6fa9216d1214f3db7.tar.gz	

bpf, riscv: Use compressed instructions in the rv64 JIT

```
This patch uses the RVC support and encodings from bpf_jit.h to optimize
the rv64 jit.

The optimizations work by replacing emit(rv_X(...)) with a call to a
helper function emit_X, which will emit a compressed version of the
instruction when possible, and when RVC is enabled.

The JIT continues to pass all tests in lib/test_bpf.c, and introduces
no new failures to test_verifier; both with and without RVC being enabled.

Most changes are straightforward replacements of emit(rv_X(...), ctx)
with emit_X(..., ctx), with the following exceptions bearing mention;

* Change emit_imm to sign-extend the value in "lower", since the
checks for RVC (and the instructions themselves) treat the value as
signed. Otherwise, small negative immediates will not be recognized as
encodable using an RVC instruction. For example, without this change,
emit_imm(rd, -1, ctx) would cause lower to become 4095, which is not a
6b int even though a "c.li rd, -1" instruction suffices.

* For {BPF_MOV,BPF_ADD} BPF_X, drop using addiw,addw in the 32-bit
cases since the values are zero-extended into the upper 32 bits in
the following instructions anyways, and the addition commutes with
zero-extension. (BPF_SUB BPF_X must still use subw since subtraction
does not commute with zero-extension.)
```

RV64 BPF 异常处理表支持

- 简化内存读取逻辑
  - bpf_probe_read(dst, size, ptr_of_val)
  - dst = ptr_of_val
- 保证内存读取安全
  - fixup when illegal memory access
  - 防止内核崩溃



author	Tong Tiangen <tongtiangen@huawei.com>	2021-10-27 11:18:22 +0000
committer	Daniel Borkmann <daniel@iogearbox.net>	2021-10-28 01:02:44 +0200
commit	252c765bd764a246a8bd516fabf6d6123df4a24f (patch)	
tree	47ba312f3ec4698cd0829237b01d3f2bbc5ee492 /arch/riscv/net	
parent	03e6a7a94001b9582ef6549e5709f3d684217b28 (diff)	
download	linux-252c765bd764a246a8bd516fabf6d6123df4a24f.tar.gz	

riscv, bpf: Add BPF exception tables

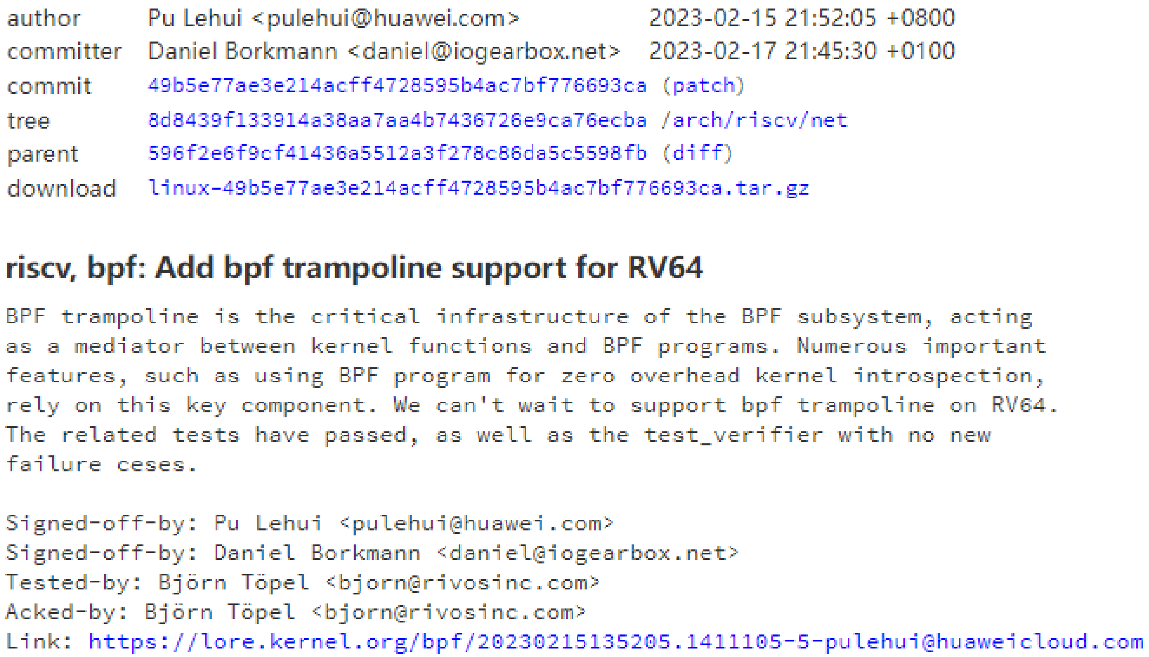
```
When a tracing BPF program attempts to read memory without using the
bpf_probe_read() helper, the verifier marks the load instruction with
the BPF_PROBE_MEM flag. Since the riscv JIT does not currently recognize
this flag it falls back to the interpreter.

Add support for BPF_PROBE_MEM, by appending an exception table to the
BPF program. If the load instruction causes a data abort, the fixup
infrastructure finds the exception table and fixes up the fault, by
clearing the destination register and jumping over the faulting
instruction.

A more generic solution would add a "handler" field to the table entry,
like on x86 and s390. The same issue in ARM64 is fixed in 800834285361
("bpf, arm64: Add BPF exception tables").
```

RV64 BPF trampoline

- 零开销Tracing
  - 类似Kprobe/Kretprobe
  - fentry, fexit

- 安全拦截
  - LSM HOOK
  - fmod_ret

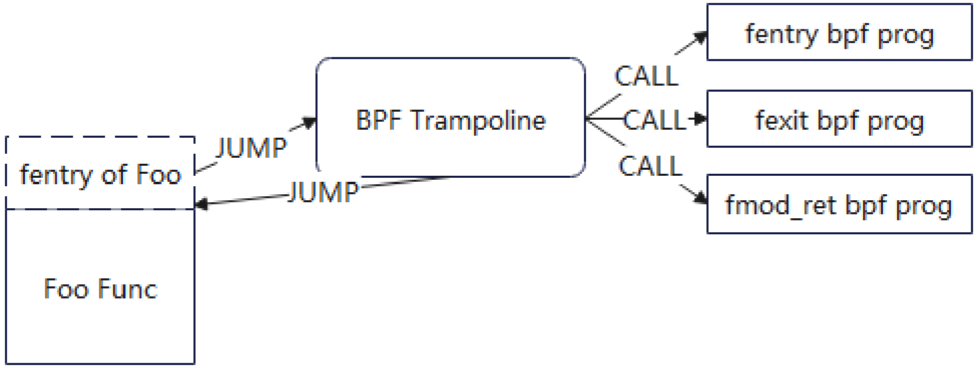
- 可编程扩展
  - 拥塞算法优化、sched_ext
  - struct_ops



```
author      Pu Lehui <pulehui@huawei.com>                  2023-02-15 21:52:05 +0800
committer   Daniel Borkmann <daniel@iogearbox.net>         2023-02-17 21:45:30 +0100
commit      49b5e77ae3e214acff4728595b4ac7bf776693ca (patch)
tree        8d8439f133914a38aa7aa4b7436726e9ca76ecba /arch/riscv/net
parent      596f2e6f9cf41436a5512a3f278c86da5c5598fb (diff)
download    linux-49b5e77ae3e214acff4728595b4ac7bf776693ca.tar.gz
```

riscv, bpf: Add bpf trampoline support for RV64

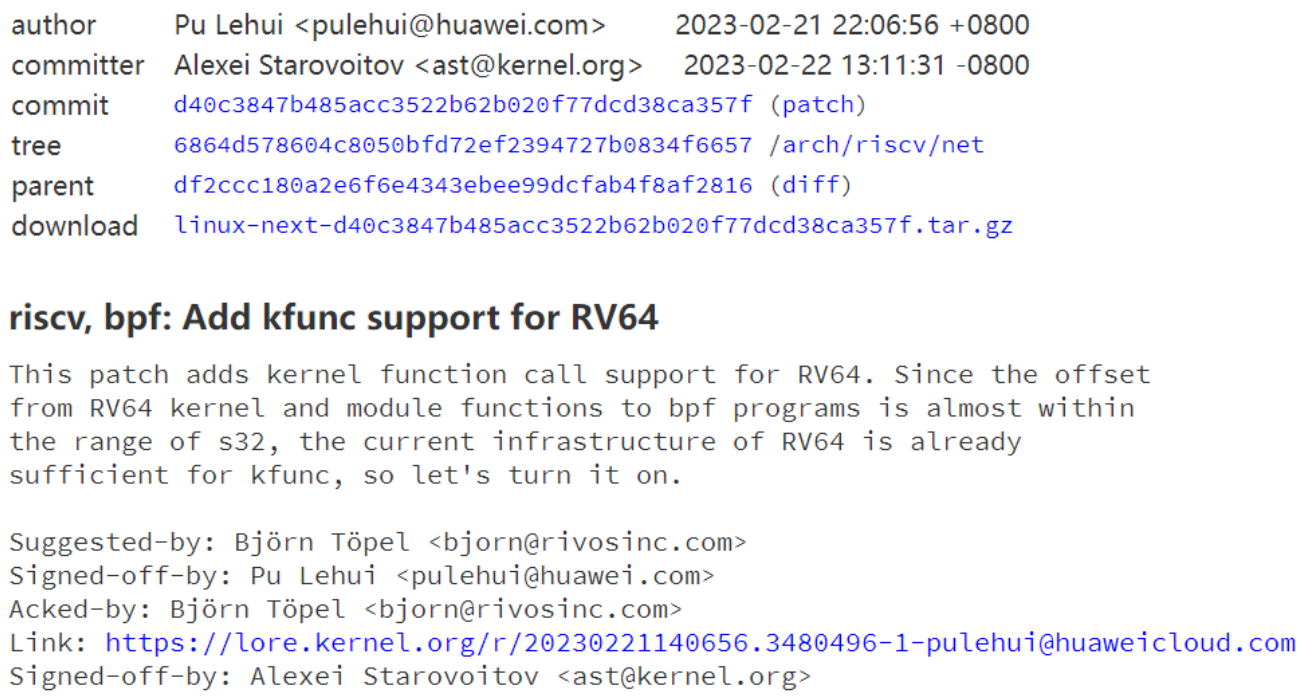
```
BPF trampoline is the critical infrastructure of the BPF subsystem, acting
as a mediator between kernel functions and BPF programs. Numerous important
features, such as using BPF program for zero overhead kernel introspection,
rely on this key component. We can't wait to support bpf trampoline on RV64.
The related tests have passed, as well as the test_verifier with no new
failure ceses.

Signed-off-by: Pu Lehui <pulehui@huawei.com>
Signed-off-by: Daniel Borkmann <daniel@iogearbox.net>
Tested-by: Björn Töpel <bjorn@rivosinc.com>
Acked-by: Björn Töpel <bjorn@rivosinc.com>
Link: https://lore.kernel.org/bpf/20230215135205.1411105-5-pulehui@huaweicloud.com
```



RV64 BPF kfunc

- 支持调用kernel function
  - bpf helper：新增复杂、ABI固定
  - 依赖BTF of kernel function

- 优点
  - 定制化（kernel、module）
  - 易用性



author	Pu Lehui <pulehui@huawei.com>	2023-02-21 22:06:56 +0800
committer	Alexei Starovoitov <ast@kernel.org>	2023-02-22 13:11:31 -0800
commit	d40c3847b485acc3522b62b020f77dcd38ca357f (patch)	
tree	6864d578604c8050bfd72ef2394727b0834f6657 /arch/riscv/net	
parent	df2ccc180a2e6f6e4343ebee99dcfab4f8af2816 (diff)	
download	linux-next-d40c3847b485acc3522b62b020f77dcd38ca357f.tar.gz	

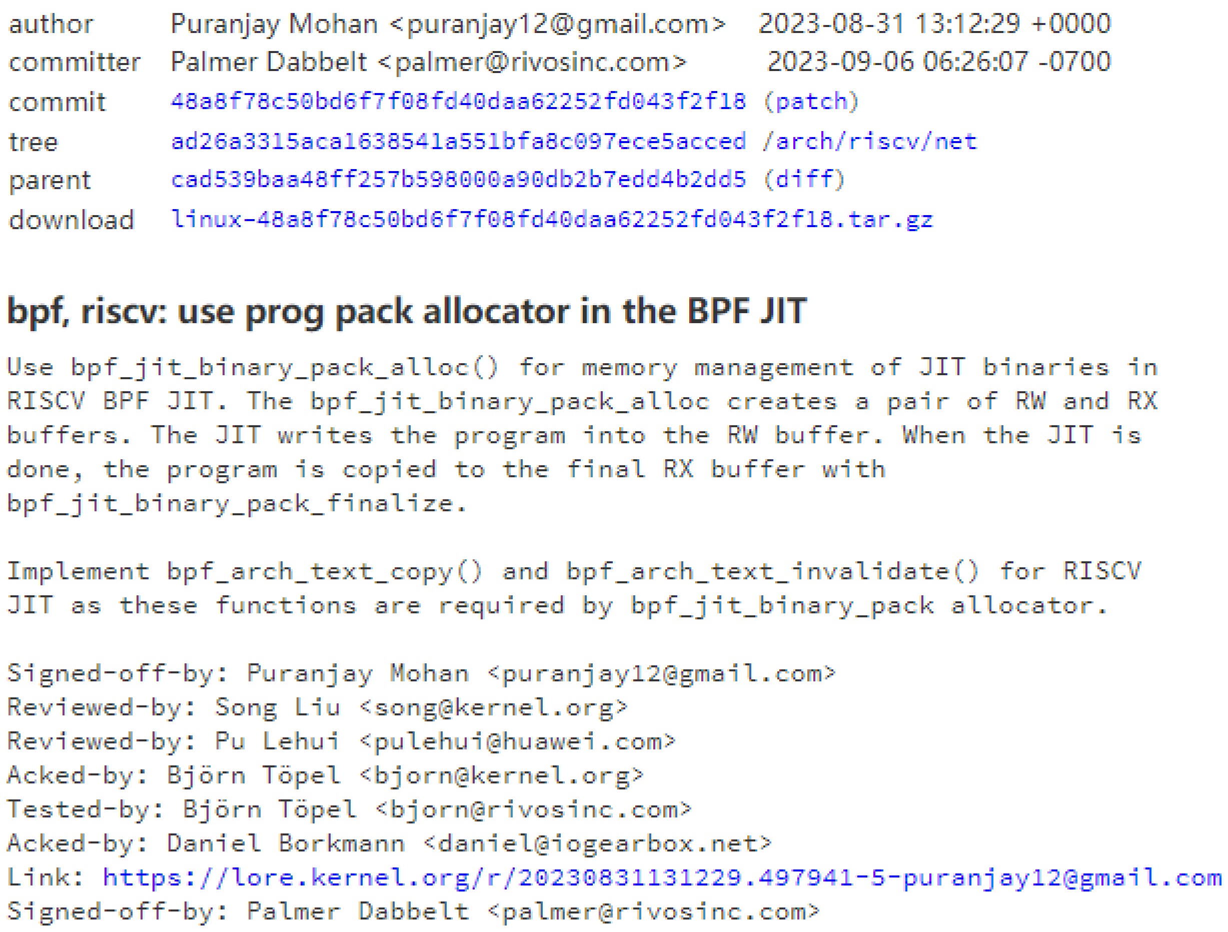
riscv, bpf: Add kfunc support for RV64

```
This patch adds kernel function call support for RV64. Since the offset
from RV64 kernel and module functions to bpf programs is almost within
the range of s32, the current infrastructure of RV64 is already
sufficient for kfunc, so let's turn it on.

Suggested-by: Björn Töpel <bjorn@rivosinc.com>
Signed-off-by: Pu Lehui <pulehui@huawei.com>
Acked-by: Björn Töpel <bjorn@rivosinc.com>
Link: https://lore.kernel.org/r/20230221140656.3480496-1-pulehui@huaweicloud.com
Signed-off-by: Alexei Starovoitov <ast@kernel.org>
```

RV64 BPF pack allocator

- 背景
  - 系统存在大量bpf程序
  - iTLB miss严重

- 实现
  - 2M大页
  - 聚拢小体积bpf程序



author	Puranjay Mohan <puranjay12@gmail.com>	2023-08-31 13:12:29 +0000
committer	Palmer Dabbelt <palmer@rivosinc.com>	2023-09-06 06:26:07 -0700
commit	48a8f78c50bd6f7f08fd40daa62252fd043f2f18 (patch)	
tree	ad26a3315aca1638541a551bfa8c097ece5acced /arch/riscv/net	
parent	cad539baa48ff257b598000a90db2b7edd4b2dd5 (diff)	
download	linux-48a8f78c50bd6f7f08fd40daa62252fd043f2f18.tar.gz	

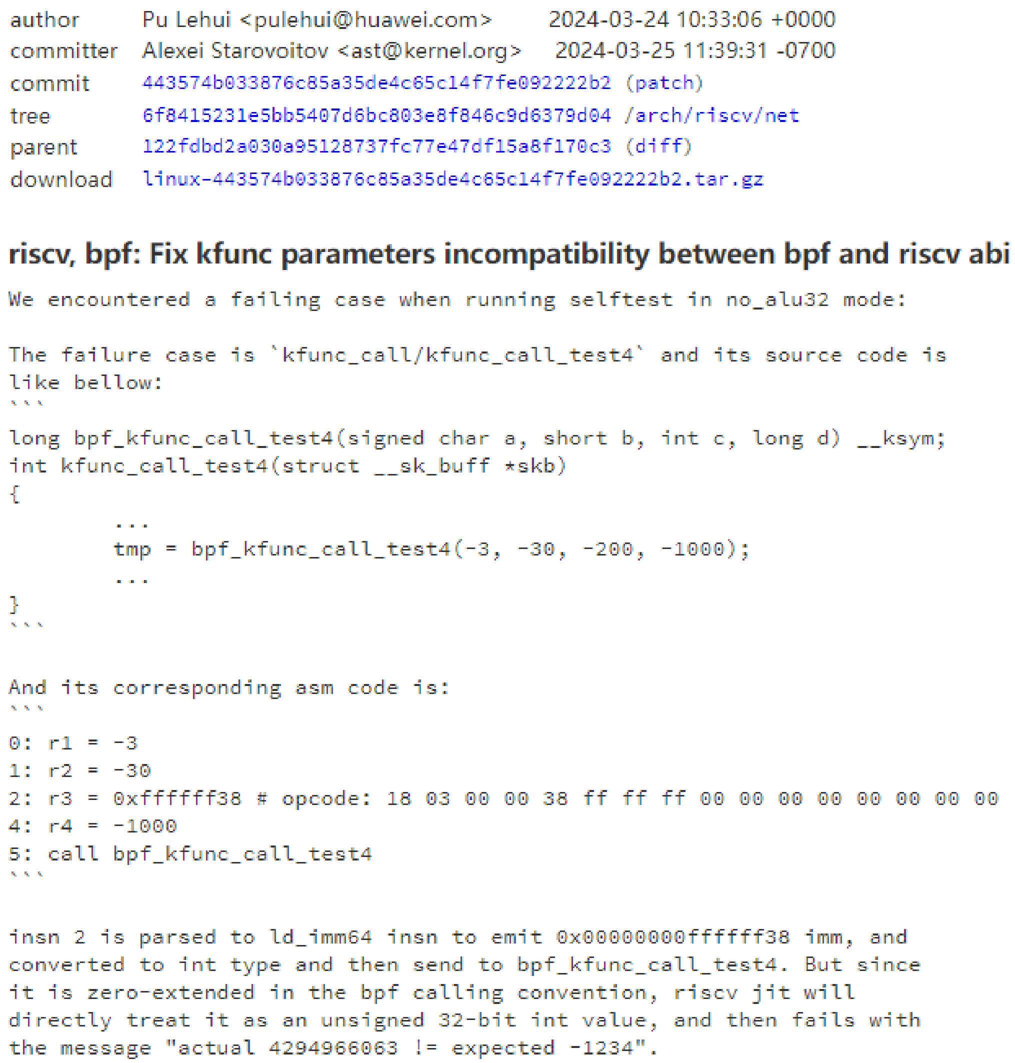
bpf, riscv: use prog pack allocator in the BPF JIT

```
Use bpf_jit_binary_pack_alloc() for memory management of JIT binaries in
RISCV BPF JIT. The bpf_jit_binary_pack_alloc creates a pair of RW and RX
buffers. The JIT writes the program into the RW buffer. When the JIT is
done, the program is copied to the final RX buffer with
bpf_jit_binary_pack_finalize.

Implement bpf_arch_text_copy() and bpf_arch_text_invalidate() for RISCV
JIT as these functions are required by bpf_jit_binary_pack allocator.

Signed-off-by: Puranjay Mohan <puranjay12@gmail.com>
Reviewed-by: Song Liu <song@kernel.org>
Reviewed-by: Pu Lehui <pulehui@huawei.com>
Acked-by: Björn Töpel <bjorn@kernel.org>
Tested-by: Björn Töpel <bjorn@rivosinc.com>
Acked-by: Daniel Borkmann <daniel@iogearbox.net>
Link: https://lore.kernel.org/r/20230831131229.497941-5-puranjay12@gmail.com
Signed-off-by: Palmer Dabbelt <palmer@rivosinc.com>
```

RV64 修复BPF和RISC-V ABI兼容

- 背景
  - BPF：对32位imm进行零扩展
  - riscv：对32位imm进行符号扩展
  - kfunc call：影响
  - helper call：不影响，总是u64

- 实现
  - 对kfunc call的32-bit入参进行符号扩展



author	Pu Lehui <pulehui@huawei.com>	2024-03-24 10:33:06 +0000
committer	Alexei Starovoitov <ast@kernel.org>	2024-03-25 11:39:31 -0700
commit	443574b033876c85a35de4c65c14f7fe092222b2 (patch)	
tree	6f8415231e5bb5407d6bc803e8f846c9d6379d04 /arch/riscv/net	
parent	122fdbd2a030a95128737fc77e47df15a8f170c3 (diff)	
download	linux-443574b033876c85a35de4c65c14f7fe092222b2.tar.gz	

riscv, bpf: Fix kfunc parameters incompatibility between bpf and riscv abi

~~~~
We encountered a failing case when running selftest in no_alu32 mode:

The failure case is `kfunc_call/kfunc_call_test4` and its source code is
like bellow:
```
long bpf_kfunc_call_test4(signed char a, short b, int c, long d) __ksym;
int kfunc_call_test4(struct __sk_buff *skb)
{
        ...
        tmp = bpf_kfunc_call_test4(-3, -30, -200, -1000);
        ...
}
```

And its corresponding asm code is:
```
0: r1 = -3
1: r2 = -30
2: r3 = 0xffffff38 # opcode: 18 03 00 00 38 ff ff ff 00 00 00 00 00 00 00 00
4: r4 = -1000
5: call bpf_kfunc_call_test4
```

insn 2 is parsed to ld_imm64 insn to emit 0x00000000ffffff38 imm, and
converted to int type and then send to bpf_kfunc_call_test4. But since
it is zero-extended in the bpf calling convention, riscv jit will
directly treat it as an unsigned 32-bit int value, and then fails with
the message "actual 4294966063 != expected -1234".
~~~~

03 RISC-V eBPF 规划

RISC-V eBPF 规划

- More Alignment
  - new features
  - exceptions, mixing tailcalls, etc.

- More Extensions
  - optimization, functionality
  - RVA22/RVA23
    https://github.com/riscv/riscv-profiles/blob/main/rva23-profile.adoc

- More Robust
  - bugfix
  - refactor

- More Fast
  - insns count reduce
  - new algorithm/mechanism
  - rely on real hardware (VisionFive2)

Thank you

群聊：CLK_2024_华为技术分享群

该二维码7天内(10月28日前)有效，重新进入将更新

APPENDIX

- 一键RV64 BPF运行环境
https://github.com/pulehui/riscv-bpf-vmtest

- Specifications
https://riscv.org/technical/specifications/