



SFrame for Arm64 Reliable Stacktrace

Dylan Hatch <dylanbhatch@google.com>

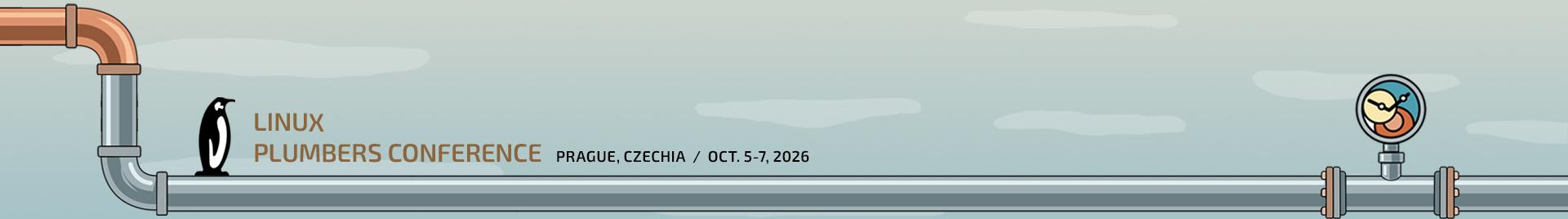


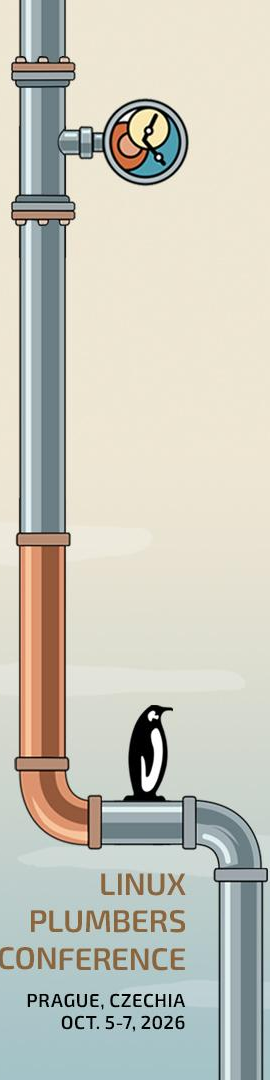
LINUX
PLUMBERS
CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026

Last Year's Slides

<https://lpc.events/event/19/contributions/2077/>





Agenda

- Reliable Stacktrace and the Livepatch Consistency Model
- Providing Reliable Stacktrace
- Current Reliable Stacktrace/Livepatch Support for Arm64
- SFrame as a Solution
- Open Questions

LINUX
PLUMBERS
CONFERENCE

PRAGUE, CZECHIA
OCT. 5-7, 2026

The Livepatch Consistency Model

Upon enabling a patch, the Livepatch subsystem enters a transition state

- Individual tasks switch over to the patched state when confirmed not to be executing any affected functions
- Transition state only ends when all tasks have been switched to the target patch state

Refs

<https://docs.kernel.org/livepatch/livepatch.html#consistency-model>



**LINUX
PLUMBERS CONFERENCE**

PRAGUE, CZECHIA / OCT. 5-7, 2026



The Livepatch Consistency Model

Livepatch applies three distinct strategies to enforce its consistency model. The first two are partial solutions that do not rely on stacktracing:

- Transition a task when it returns to user space from a system call/user space IRQ/signal
- Call `klp_update_patch_state()` on current task directly

Refs

<https://docs.kernel.org/livepatch/livepatch.html#consistency-model>



LINUX

PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026



The Livepatch Consistency Model

The third strategy is essential for livepatch: **Check the stacks of sleeping tasks. If no patched functions are present, mark it as patched.**

- The most effective strategy to apply patches
- Needs `HAVE_RELIABLE_STACKTRACE`
- Architectures without `HAVE_RELIABLE_STACKTRACE` are not fully supported for livepatch.

Refs

<https://docs.kernel.org/livepatch/livepatch.html#consistency-model>



LINUX
PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026



Providing Reliable Stacktrace

Considerations for `HAVE_RELIABLE_STACKTRACE`:

- Successful (complete) termination of trace
- Interrupts/exceptions in stack
- Ftrace and kprobe trampoline
- Link register/return address unreliability

Refs

<https://docs.kernel.org/livepatch/reliable-stacktrace.html>



LINUX
PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026



Current Support on Arm64

A v6.12 patch series landed adds the concept of a “Meta” frame record to explicitly identify key points of interest on the stack:

- Exception boundaries – `FRAME_META_TYPE_PT_REGS`
- Final frames – `FRAME_META_TYPE_FINAL`

Marks the two missing pieces to meeting the `HAVE_RELIABLE_STACKTRACE` requirements.

Refs

<https://lore.kernel.org/linux-arm-kernel/20241017092538.1859841-1-mark.rutland@arm.com/>

Mark Rutland <mark.rutland@arm.com>



LINUX
PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026



Current Support on Arm64

Limitation: The link register is not always live when an exception occurs

- LR unwinding has to be skipped, meaning unwinding is still unreliable at the exception boundary
- **HAVE_RELIABLE_STACKTRACE** is supported by stopping at exception boundaries when the trace needs to be reliable
- Exception boundaries are detected but not (yet) reliably unwound

Refs

<https://lore.kernel.org/all/20241211140704.2498712-2-mark.rutland@arm.com/>
<https://lore.kernel.org/live-patching/20250320171559.3423224-1-song@kernel.org/>

Mark Rutland <mark.rutland@arm.com>

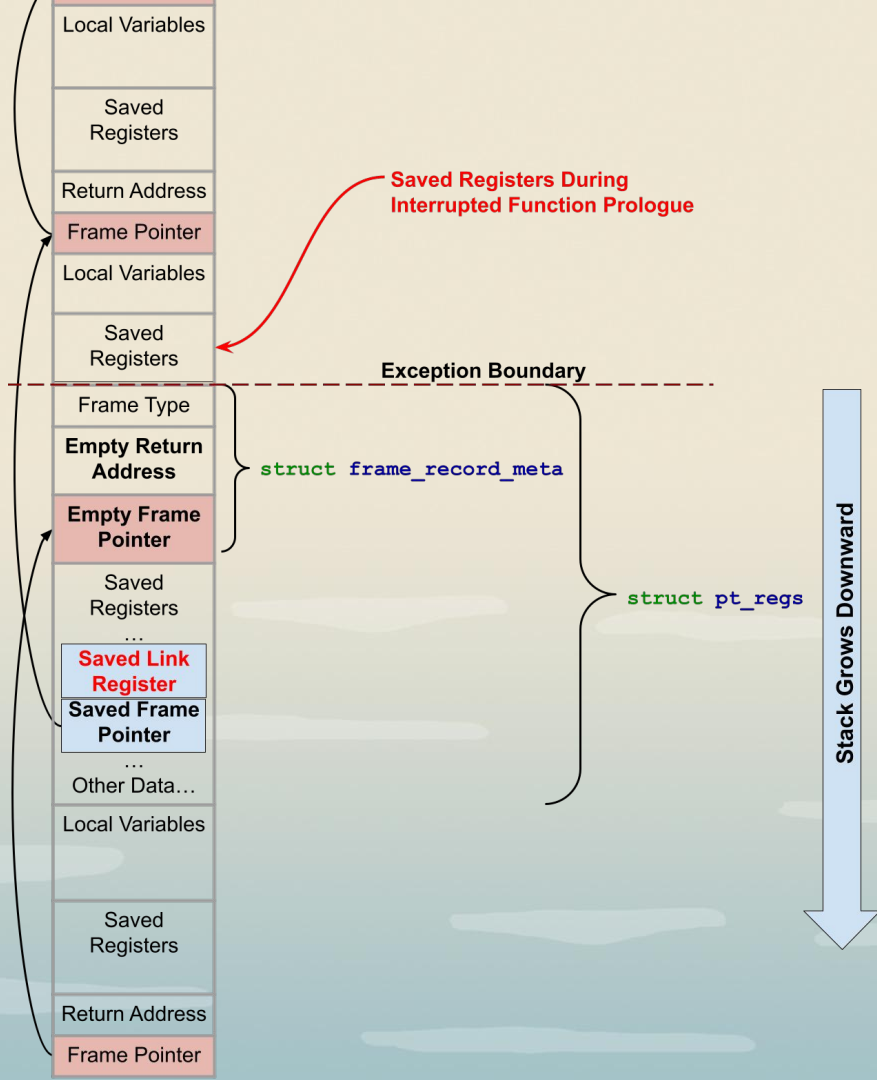
Song Liu <song@kernel.org>



LINUX
PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026



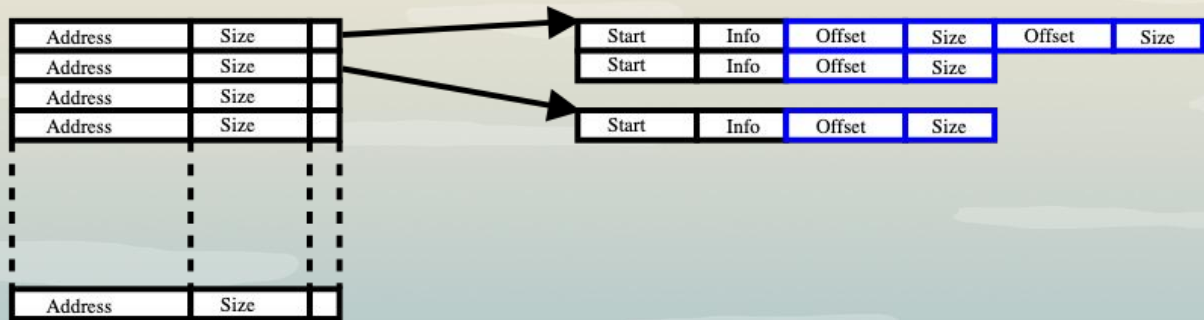


How do we know if we should use the Link Register?

SFrame as a Solution

SFrame provides a generalized, toolchain-based solution, inspired by the ORC unwinder

- Provides an assembler-generated .sframe section, containing a two-level lookup structure without reverse-engineering the binary



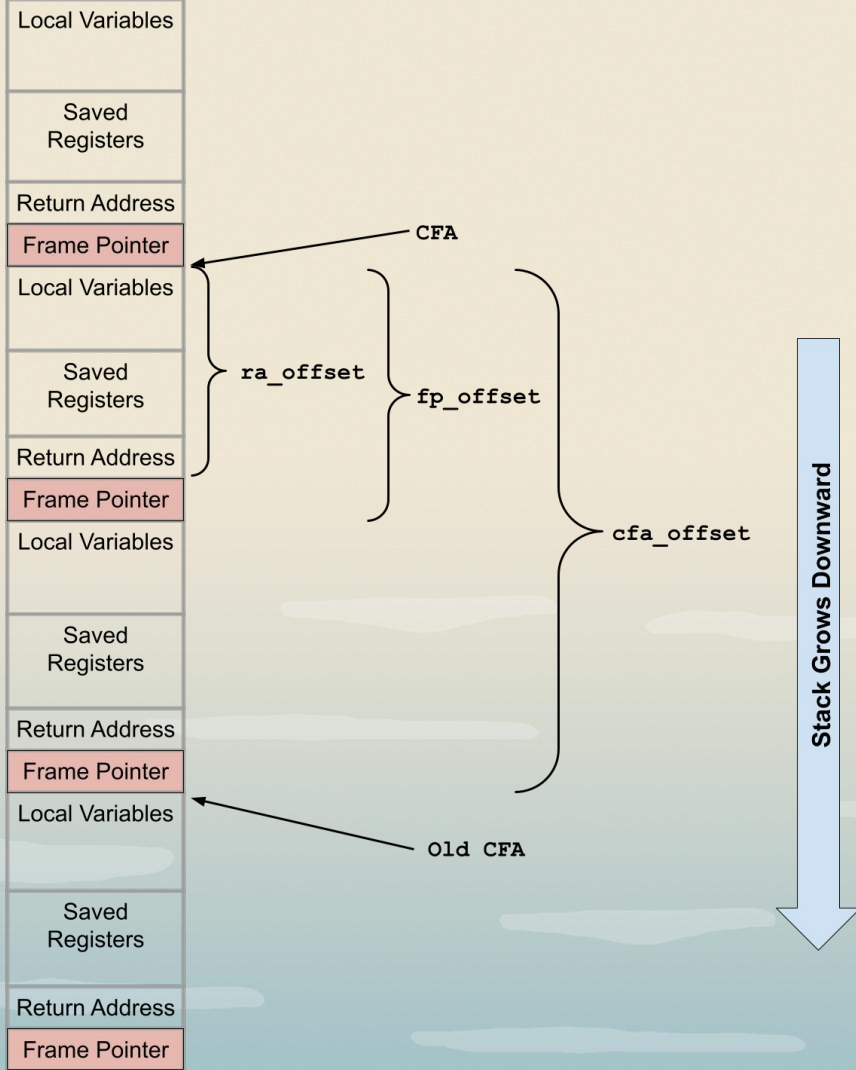
Refs

<https://lwn.net/Articles/1029189/>

<https://sourceware.org/binutils/docs/sframe-spec.html>

https://static.sched.com/hosted_files/ossna2023/dl/OSS_LinuxCon_SFrame_final.pdf?gl=1

Indu Bhagat <indu.bhagat@oracle.com>



Provides the minimal necessary information to provide a stacktrace:

- Frame Pointer (FP)
- Return Address (RA)
- Canonical Frame Address (CFA)

SFrame as a Solution

New Arm64 stacktrace logic:

- In most cases, continue to unwind with Frame Pointer
- After recovering a frame from `struct pt_regs`, recover the following frame with an SFrame lookup

Refs

<https://lore.kernel.org/all/20261007114335.440322-1-dylanbhatch@google.com/>

Dylan Hatch <dylanbhatch@google.com>

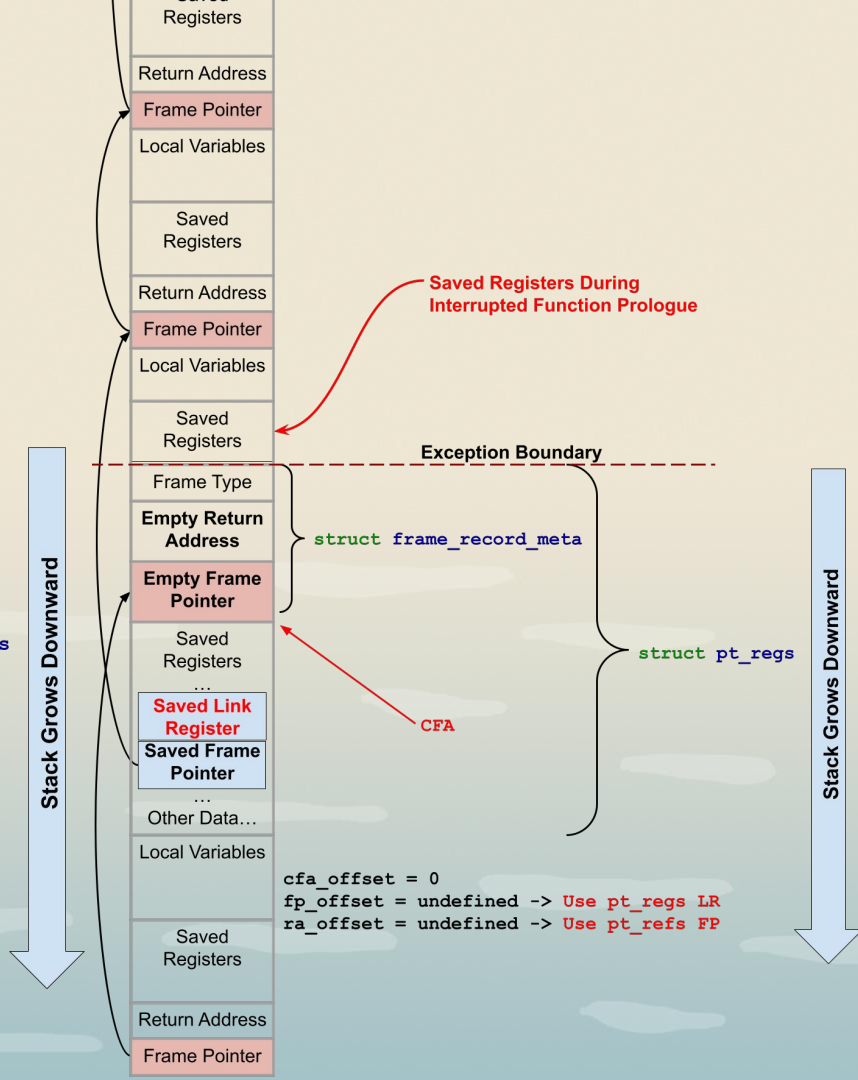
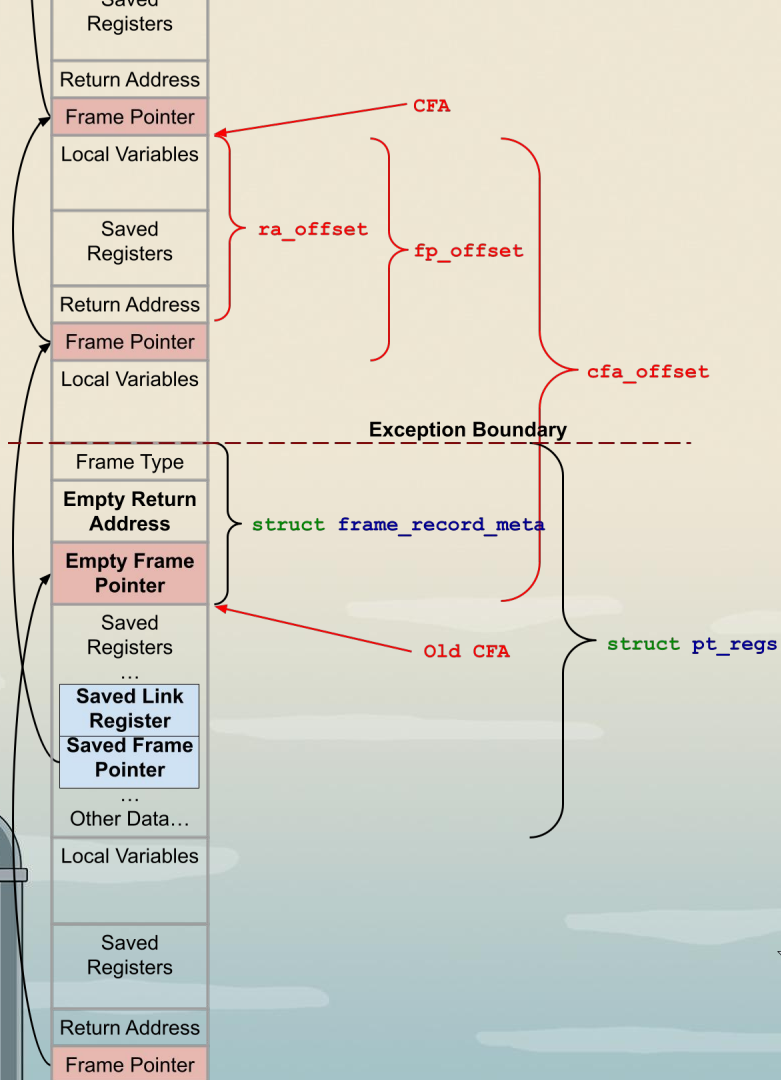
Retain the performance benefit of Frame Pointers, with added reliability of SFrame



LINUX
PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026





SFrame as a Solution

Please review, test, etc

Refs

<https://lore.kernel.org/all/20261007114335.440322-1-dylanbhatch@google.com/>

Dylan Hatch <dylanbhatch@google.com>



LINUX
PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026



Open Questions (1/2)

SFrame is currently only supported in GCC. Given the significant pushback on the SFrame format, is there a pathway for supporting in LLVM?

For the specific purpose of handling exception boundaries, is there an alternative approach for determining link register currency?

Jens Remus has sent an RFC for an `.eh_frame` unwinder in userspace. Given this, is there any remaining prospect for a userspace SFrame unwinder?

Refs

<https://discourse.llvm.org/t/rfc-adding-sframe-support-to-llvm/86900>

<https://lore.kernel.org/all/20260505121718.3572346-1-jremus@linux.ibm.com/>

<https://lore.kernel.org/all/20260821195259.2688377-1-jremus@linux.ibm.com/>

Jens Remus <jremus@linux.ibm.com>

Josh Poimboeuf <jpoimboe@kernel.org>

Steven Rostedt <rostedt@kernel.org>



LINUX

PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026



Open Questions (2/2)

Handwritten assembly requires CFI directives to reliably unwind. These patches add directives to known leaf functions, but skip other assembly functions. Is there a better / more maintainable way to do this?

What is the potential for future work for SFrame in the kernel?
Supporting SFrame-only unwinder (no Frame Pointer) on Arm64?
Replacing ORC on X86? Other architectures?

What's the best way to generate .sframe for livepatches in KLP-Build?

Refs

<https://discourse.llvm.org/t/rfc-adding-sframe-support-to-llvm/86900>

<https://lore.kernel.org/all/20260505121718.3572346-1-jremus@linux.ibm.com/>

<https://lore.kernel.org/all/20260821195259.2688377-1-jremus@linux.ibm.com/>

Jens Remus <jremus@linux.ibm.com>

Josh Poimboeuf <jpoimboe@kernel.org>

Steven Rostedt <rostedt@kernel.org>



LINUX

PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026

