Progress on BTF linking and deduplication using the toolchain

Nick Alcock
Bruce McCulloch

LSF/MM 04 May 2026

Overview

CTFv4

- Overview
- Motivation
- CTFv4 is BTF-compatible
- New in CTFv4
- libctf changes since Cauldron
- Archive & Linking Changes

pahole & kernel

- Overview
- GCC generated BTF
- Linking (vmlinux & modules)
- pahole changes
- Compatibility
- Transition & Maintenance

Motivation

- Widespread availability of type info in userland as well as the kernel
- For userspace, it can work automatically; the kernel is a bit of a special case, hence pahole
- (Ultimately, for userland) automatic runtime ABI conflict detection
- But at this stage we’re just trying to dedup the kernel using toolchain-generated BTF

CTFv4

CTF is BTF-compatible

- CTFv4 is a strict superset of BTF (header and all)
- All valid BTF is also valid CTFv4
- Kind layout section support will let BTF readers skip new CTF type kinds (userspace floats, etc), while still reading the rest; remaining incompatibilities will be eliminated with vlen and kind resizings
- CTFv4 is more expressive for userspace but the expressiveness is in areas users of BTF aren't likely to care about. The additions largely benefit userspace (ELF symbol lookup, etc). Done via extra ELF sections, so BTF compat is unaffected.
- Split BTF-in-one-file (useful for userspace section storage) is done via a suggestion from Alexei: straight concatenation!

libctf changes since Cauldron

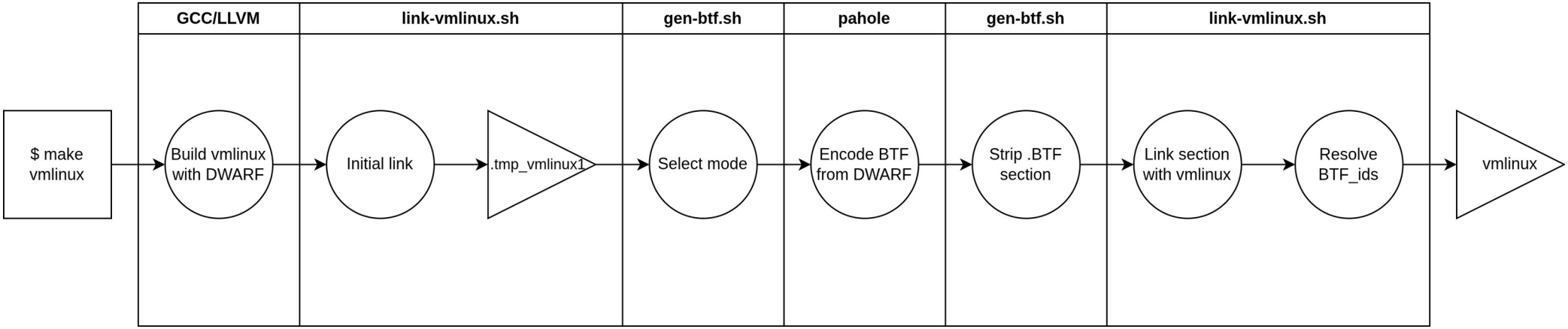
- libctf can read, emit, dedup, and query both BTF and CTF, and convert them to each other
- libctf prefers to write out BTF unless doing so would cause data loss
- Emission of particular kinds can be restricted (erroring) or replaced with BTF_KIND_UNKNOWN
- Massive API simplifications (easy to transition to)
- Both C-type-system querying functions and the ability to just get raw BTF data for any type

Archive and Linking Changes

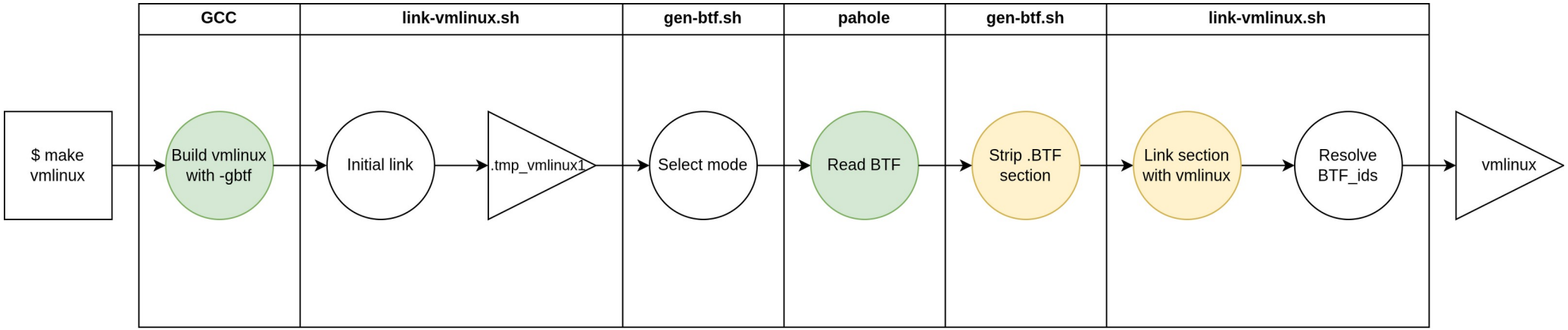
- libctf now writes out Alexei's new concatenation-based BTF archive format
  - Old format still readable
- The libctf deduplicator (as used by ld) puts ambiguously-defined types in child dicts named after the CU
- BTF has no header fields for CU names, and none will be accepted
  - Solved by putting an array of cuname strings *after the archive*, preceded by a magic number: nobody but libctf ever needs to know about it
- libctf works without this being present: thus, compatible with other emitters, including non-BTF-aware linkers
  - members are treated as just being numbered
- Required us to drop compression; will bring it back

pahole & kernel

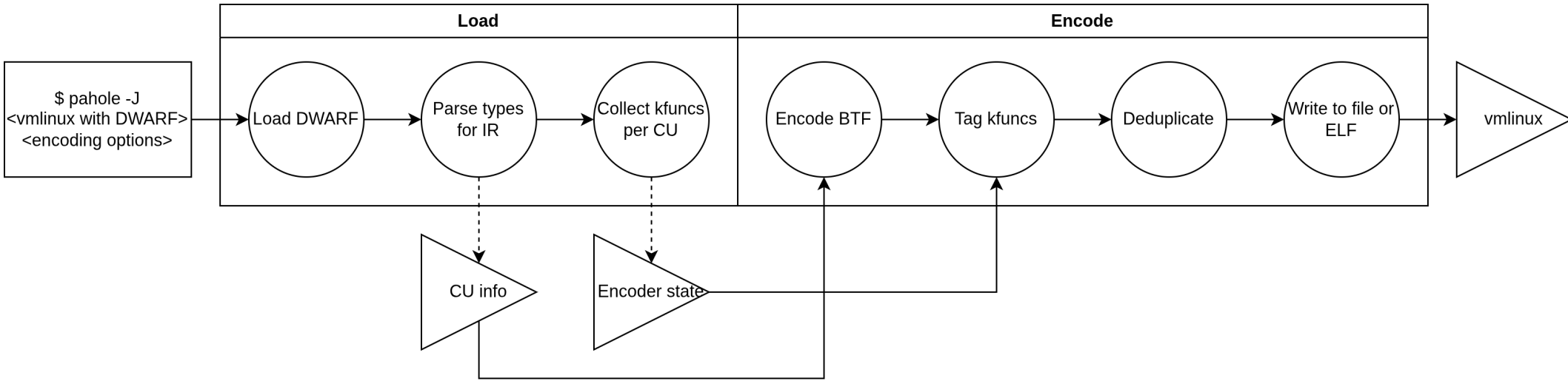
Current State of BTF Generation Overview (vmlinux)



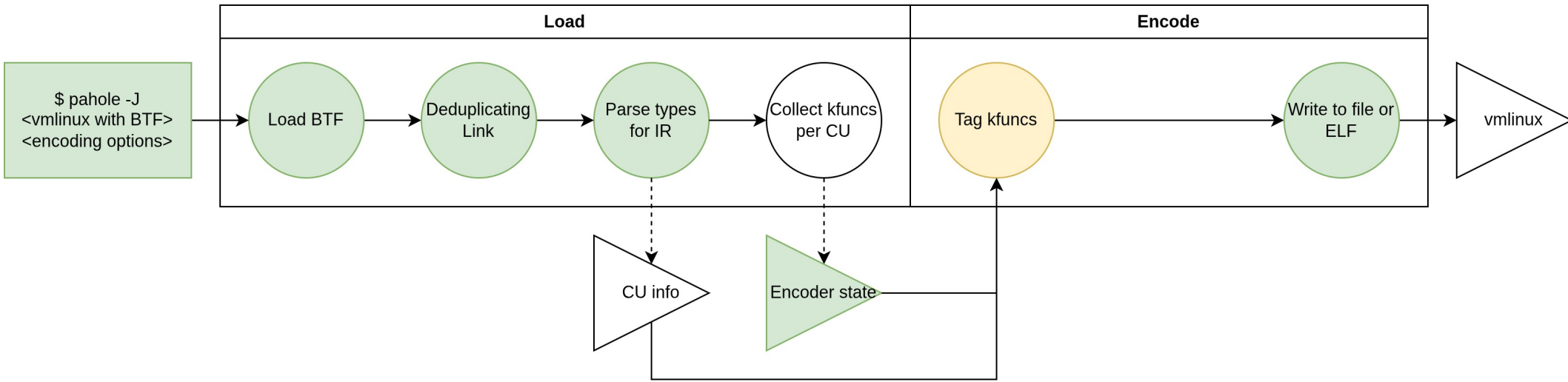
Proposed BTF Generation Overview (vmlinux)



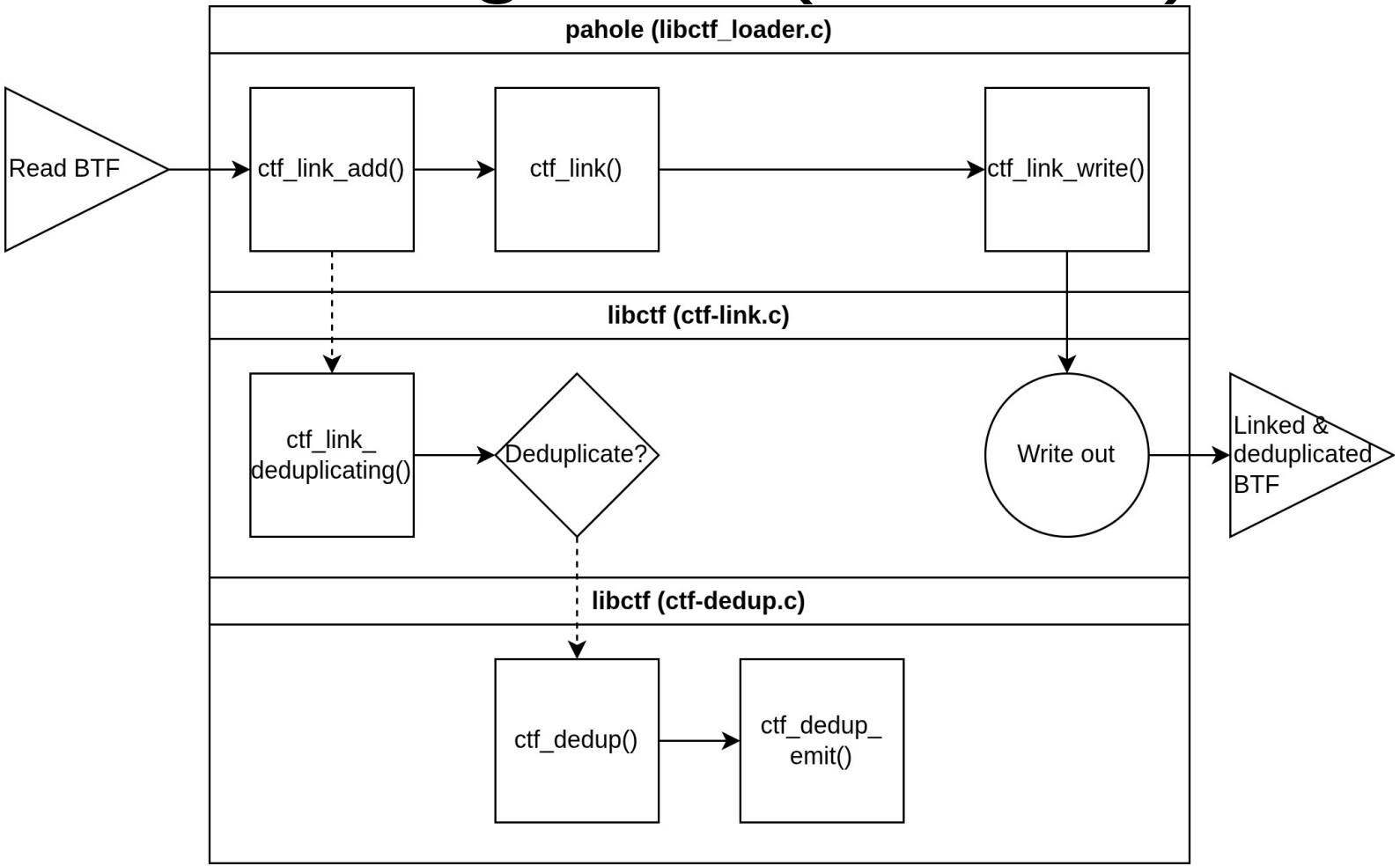
Current State of BTF Generation pahole (vmlinux)



Proposed BTF Generation pahole (vmlinux)

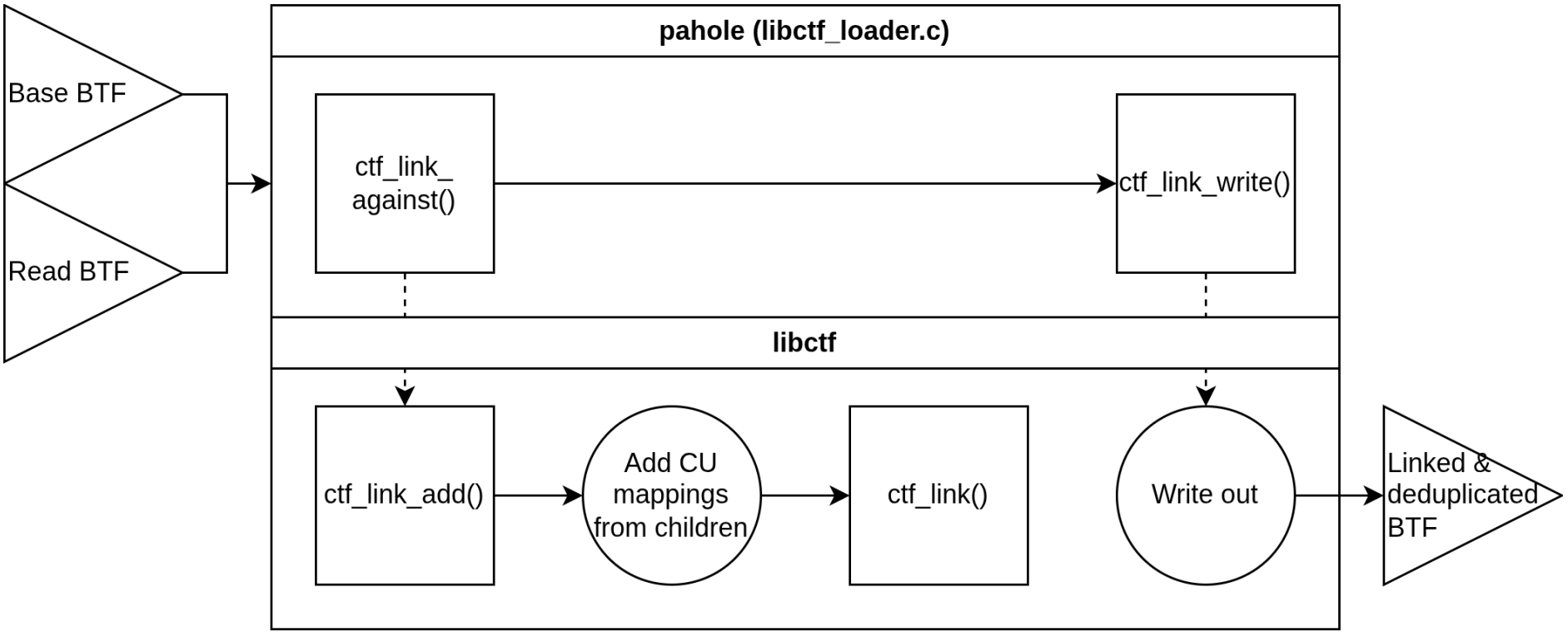


Linking BTF (vmlinux)



Linking BTF (modules)

- Special “Against Types” mode



Changes to pahole

- New loader pathway: libctf
- Modified encoding is required to handle kfuncs and writeout
- 1:1 parity with BTF loader pathway
- Removed dead Solaris CTF pathways

Compatibility

- Drop-in alternative for other loaders in pahole
- Old pathways remain if you want them
- New kernel config check:
  - CONFIG_HAVE_BTF_TOOLCHAIN
- Adding kernel config option:
  - CONFIG_USE_BTF_TOOLCHAIN

Where to find it

- GCC 16+
- binutils
  - https://sourceware.org/git/binutils-gdb.git
    users/nalcock/road-to-ctfv4
- dwarves:
  - https://github.com/peter-shoes/dwarves
    bcm/libctf
- linux:
  - https://github.com/peter-shoes/linux-gccbtf/
    libctf