Challenges with Kernels built with LTO

Song Liu
Yonghong Song

2025.12.11 @ LPC 2025

LTO vs. Tracing

Kernel function inlining and tracing

- Kernel function inlining makes tracing (ftrace, kprobe, etc.) difficult and confusing
- Why confusing? Inlined kernel functions may still exists in the kernel
  - Static functions can be inlined at some call sites
  - Without LTO, global functions can still be inlined at call sites in the same file
  - With LTO, static and global functions can also be inlined at calls sites in different files
- 6.18 kernel + llvm 20.1.8
  - Without LTO, 6895 inlined-but-still-exists kernel functions
  - With LTO, 8058 inlined-but-still-exists kernel functions

Potential solution?

- Add options to disable inlined-but-still-exists kernel functions
  - `--consistent-inlining` (?)

- If a function is inlined at some call sites, try to inline it everywhere, and remove the function
- If the above is not possible, don’t inline at all

LTO vs. Livepatch

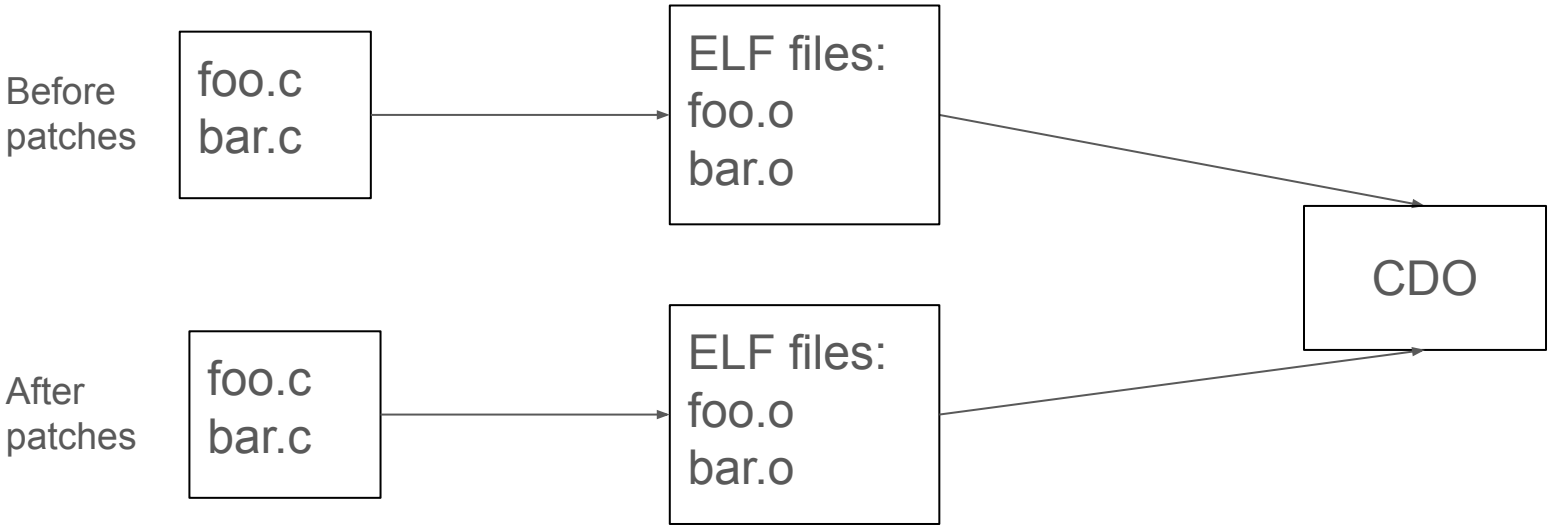
LTO and Kernel Live Patch (KLP)

- Binary diff based toolchains

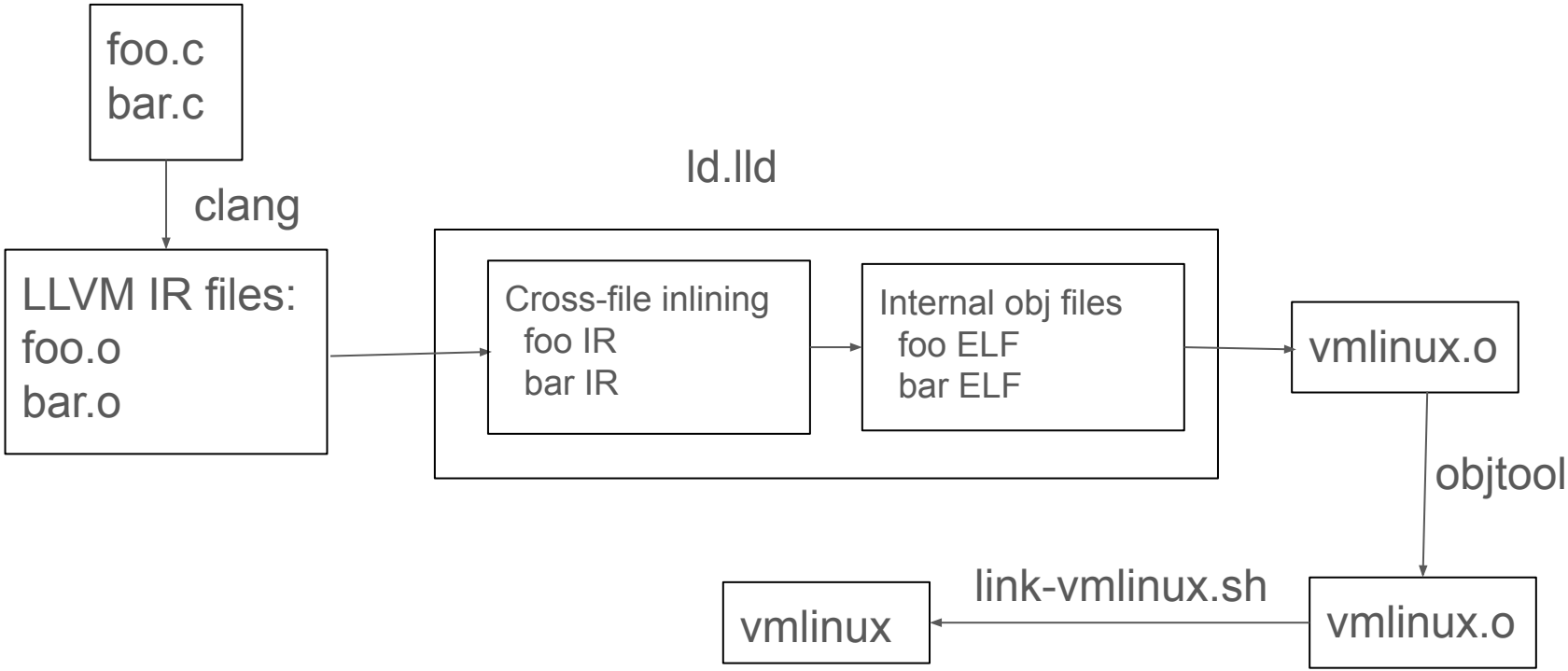
- Current toolchain: kpatch-build with downstream patches
- Also experimented with the klp-build work by Josh Poimboeuf

No-LTO Compilation Workflow

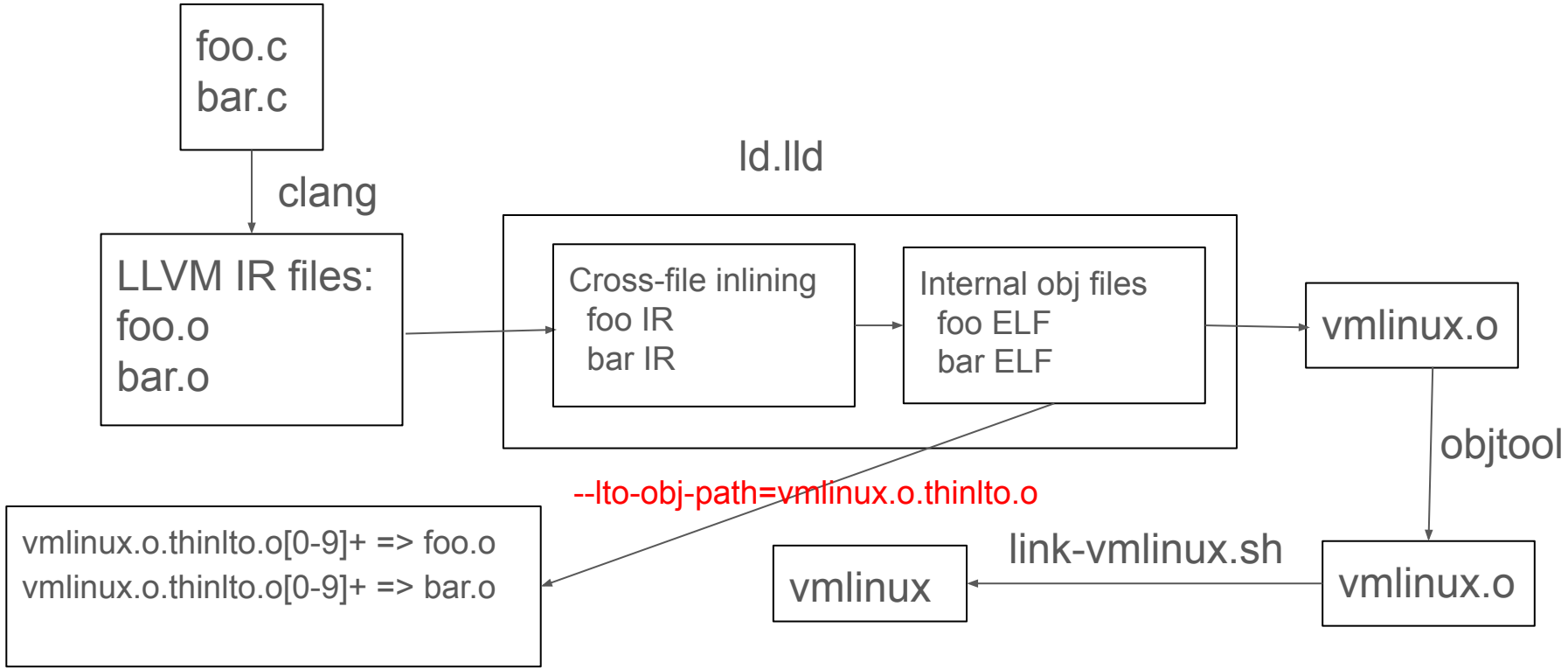
No-LTO kpatch-build Workflow



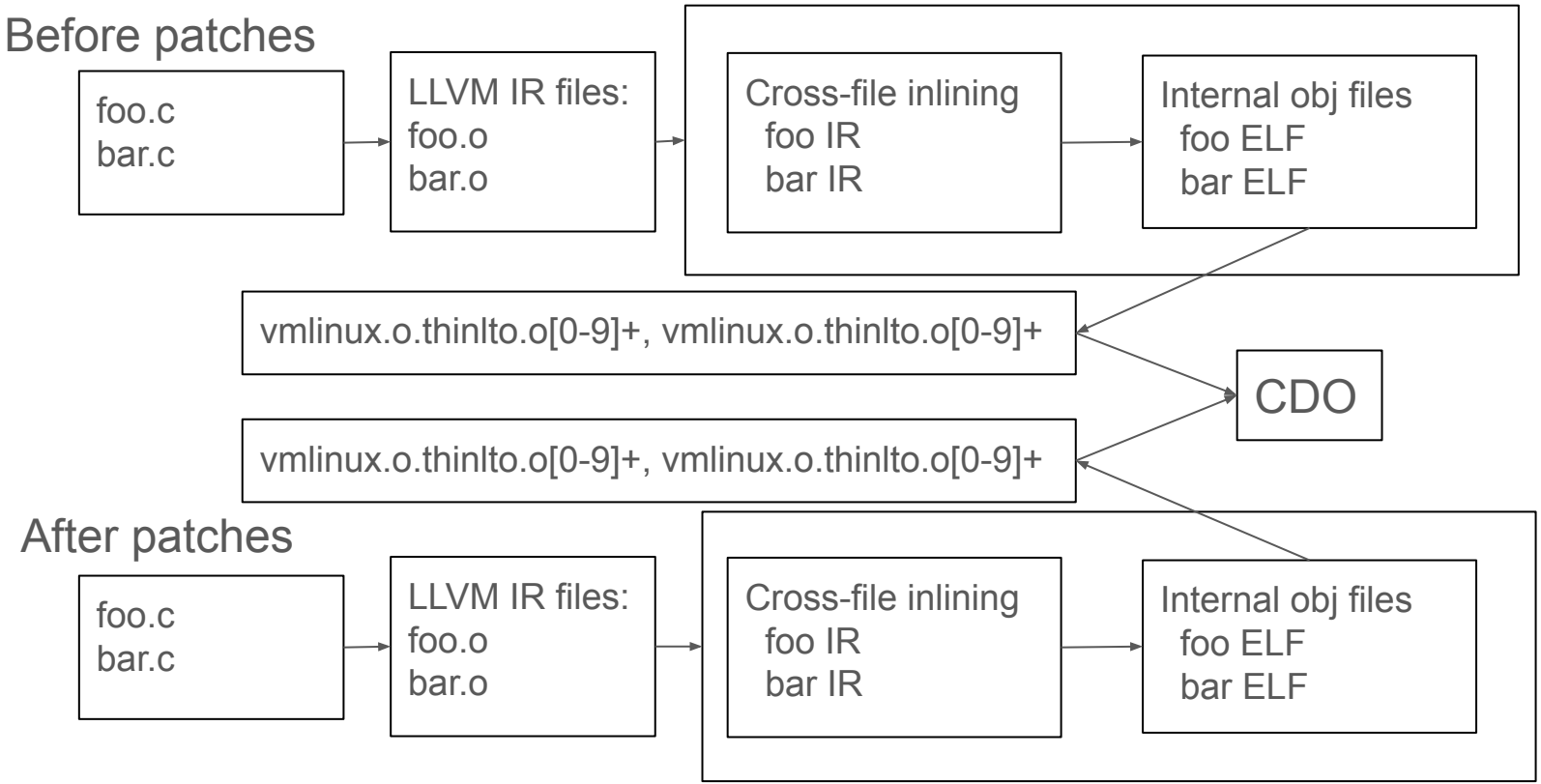
LTO Compilation Workflow



LTO Compilation Workflow



LTO kpatch-build Workflow



LTO klp-build Workflow

- Binary diff on vmlinux.o
- No need for --lto-obj-path=vmlinux.o.thinlto.o
- No arm64 support yet
- However, klp-build doesn’t fully solve the LTO challenge

Example patch

```
diff --git a/drivers/md/md.c b/drivers/md/md.c
index 41c476b40c7a..9dd7ff0b43a8 100644
--- a/drivers/md/md.c
+++ b/drivers/md/md.c
@@ -6312,6 +6312,9 @@ int md_run(struct mddev *mddev)
        struct md_personality *pers;
        bool nowait = true;

+       if (!mddev)
+               return -EINVAL;
+
        if (list_empty(&mddev->disks))
                /* cannot run an array with no devices.. */
                return -EINVAL;
```

kpatch-build log with LTO, 6.13 based kernel

```
vmlinux.o.thinlto.o2350: changed function: level_store
vmlinux.o.thinlto.o2350: changed function: md_run
vmlinux.o.thinlto.o274: changed function: devm_unregister_reboot_notifier
vmlinux.o.thinlto.o274: changed function: register_reboot_notifier
vmlinux.o.thinlto.o274: changed function: unregister_reboot_notifier
vmlinux.o.thinlto.o274: changed function: devm_register_reboot_notifier
vmlinux.o.thinlto.o274: changed function: kernel_restart
vmlinux.o.thinlto.o274: changed function: kernel_halt
vmlinux.o.thinlto.o274: changed function: kernel_power_off
vmlinux.o.thinlto.o274: changed function: kernel_restart_prepare
vmlinux.o.thinlto.o776: changed function: internal_create_groups
vmlinux.o.thinlto.o776: changed function: sysfs_create_group
vmlinux.o.thinlto.o776: changed function: sysfs_update_group
vmlinux.o.thinlto.o776: new function: internal_create_group.llvm.7469111395707979223
```

o2350: md/md.c
o274: kernel/reboot.c
o766: fs/sysfs/group.c

Why?

```
$ readelf -sW vmlinux-lto-orig | grep reboot_notifier_list
 11216: ffffffff8325fdd0    48 OBJECT  LOCAL  DEFAULT   19  reboot_notifier_list

$ readelf -sW vmlinux-lto-patched | grep reboot_notifier_list
149108: ffffffff8325fdd0    48 OBJECT  LOCAL  HIDDEN    19
reboot_notifier_list.llvm.16444911668687547466

$ readelf -rW patched/vmlinux.o.thinlto.o274
...
Relocation section '.rela.text.unlikely.kernel_restart' at offset 0x2b48 contains 16 entries:
    Offset             Info             Type               Symbol's Value  Symbol's Name + Addend
...
0000000000000011  000001060000000b R_X86_64_32S           0000000000000000
reboot_notifier_list.llvm.14714292017924923542 + 0
```

klp-build log with LTO, 6.18-rc1

```
vmlinux.o: warning: objtool: no correlation: __irf_start
vmlinux.o: warning: objtool: no correlation: __irf_end
vmlinux.o: warning: objtool: no correlation: __mddev_resume.llvm.13278040285442952784
vmlinux.o: warning: objtool: no correlation: md_submodule.llvm.13278040285442952784
vmlinux.o: error: objtool: .discard.annotate_data: missing special section entsize or annotations
vmlinux.o: new function: analyze_sbs
vmlinux.o: new function: __mddev_resume.llvm.7588846155915629702
vmlinux.o: changed function: md_start_sync
vmlinux.o: changed function: get_pers
vmlinux.o: changed function: md_bitmap_create
vmlinux.o: changed function: md_ioctl
vmlinux.o: changed function: md_exit
vmlinux.o: changed function: md_init
vmlinux.o: changed function: suspend_lo_store
vmlinux.o: changed function: suspend_hi_store
vmlinux.o: changed function: level_store
[more]
```

Why?

```
vmlinux.o: warning: objtool: no correlation: __irf_start
vmlinux.o: warning: objtool: no correlation: __irf_end
vmlinux.o: warning: objtool: no correlation: __mddev_resume.llvm.13278040285442952784
vmlinux.o: warning: objtool: no correlation: md_submodule.llvm.13278040285442952784
vmlinux.o: error: objtool: .discard.annotate_data: missing special section entsize or annotations
vmlinux.o: new function: analyze_sbs
vmlinux.o: new function: __mddev_resume.llvm.7588846155915629702
vmlinux.o: changed function: md_start_sync
vmlinux.o: changed function: get_pers
vmlinux.o: changed function: md_bitmap_create
vmlinux.o: changed function: md_ioctl
vmlinux.o: changed function: md_exit
vmlinux.o: changed function: md_init
vmlinux.o: changed function: suspend_lo_store
vmlinux.o: changed function: suspend_hi_store
vmlinux.o: changed function: level_store
[more]
```

Newer compiler appears to be more aggressive with LTO

- llvm-19.1.7 + klp-build: 26 changed functions within the same file
- llvm-20.1.8 + klp-build: a lot more changes from multiple files

Potential solutions

- Remove .llvm.<hash>. postfix when the local symbol is unique, which is 99%+ of them

- For a recent kernel with Thin-LTO
  - 2117 functions like <foo>.llvm.<hash>
  - All of them are unique without .llvm.<hash>
  - So effectively for this particular case, all .llvm.<hash> postfix can be removed.

Thank You!