Calling conventions for the lazy binding on RISC-V

PLCT实验室 陈嘉炜

jiawei@iscas.ac.cn

2021.05.26

lazy binding:

延迟绑定(lazy binding)是ELF动态链接(Dynamic Linking)的一种优化技术。

在动态链接下，程序模块直接包含大量的函数引用关系，在程序执行前，动态链接需要花费大量时间用来解决函数符号查找与重定位。但是很多函数在程序执行完成时都不一定会用到(如错误处理函数)。

所以ELF使用了lazy binding，当函数第一次被用到时才进行绑定，这样可以大幅提高程序的启动速度，尤其是一些有大量引用模块的程序。

lazy binding的实现:

```
glibc_skip_solib_resolver (struct gdbarch *gdbarch, CORE_ADDR pc)
{
  /* The GNU dynamic linker is part of the GNU C library, and is used
     by all GNU systems (GNU/Hurd, GNU/Linux).  An unresolved PLT
     entry points to "_dl_runtime_resolve", which calls "fixup" to
     patch the PLT, and then passes control to the function.

     We look for the symbol `_dl_runtime_resolve', and find `fixup' in
     the same objfile.  If we are at the entry point of `fixup', then
     we set a breakpoint at the return address (at the top of the
     stack), and continue.
```

lazy binding的问题:

H.J. Lu 2017-03-17 15:37:53 UTC

Intel C++ __regcall calling convention for x86-64:

https://software.intel.com/en-us/node/522787

passes function parameters in %xmm0-%xmm15. Since _dl_runtime_resolve only preserves %xmm0-%xmm7, %xmm8-%xmm15 may be clobbered with lazy binding.

参考链接：https://sourceware.org/bugzilla/show_bug.cgi?id=21265

lazy binding问题处理:

Florian Weimer 2017-03-20 14:05:54 UTC

Note that there is a parallel mailing list thread reviewing this ABI change proposal:

https://sourceware.org/ml/libc-alpha/2017-03/msg00343.html

I think we still need some ABI documentation even if more registers are preserved because arbitrary calling conventions still will not work. Using noplt calls as a workaround in the Intel compiler seems a reasonable fix (no ABI changes required, but this still needs documentation in the psABI supplement IMHO).

参考链接：sourceware.org/legacy-ml/libc-alpha/2017-03/msg00343.html

lazy binding on RISC-V:

问题提出：https://github.com/riscv/riscv-elf-psabi-doc/issues/66

palmer-dabbelt commented on 20 Dec 2017 — Member

See this x86 ABI bug for more details: https://sourceware.org/bugzilla/show_bug.cgi?id=21265

I vote we solve this on RISC-V by just requiring that lazily bound functions follow the standard ABI. This lets us avoid saving the rest of the X registers, and assuming we can do the fixup without any extensions (which seems reasonable, and is what we currently do despite it not actually being enforced) we won't need to save F or V state here.

I don't know where this should go in the manual, **@asb**?

RVV相关：https://github.com/riscv/riscv-elf-psabi-doc/pull/171#issuecomment-801560921

kito-cheng commented on 18 Mar — Member

I just realized this calling convention need to consider to lazy binding issue[1] too, my first impression is oh, we don't have callee-save register so we might don't have such issue, but in this proposal we have passed argument on vector register, which possible to be clobbered by lazy binding/ifunc resolver, so I guess the only possible choice for baseline vector calling convention is all argument/return values are passed in memory.

@jrtc27 **@jim-wilson** what do you think?

[1] #66

lazy binding on RISC-V:

目前解决方案：https://github.com/riscv/riscv-elf-psabi-doc/pull/190#issuecomment-841839215

kito-cheng commented 10 days ago — Member

Just note, for this changes we need to implement following component:

- assembler: New syntax for setting a symbol is STO_RISCV_VARIANT_CC.
- compiler: Must mark a symbol is STO_RISCV_VARIANT_CC if not using standard calling conversion or using parameter not passed in GPR/FPR.
- linker: generate DT_RISCV_VARIANT_CC for dynamic section if any symbol with STO_RISCV_VARIANT_CC.
- dynamic linker: glibc and all other dynamic linker must recognize this new flag and handle this.