Obstacles & Solutions for Livepatch Support on ARM64 Architecture

Li Bin (李彬) / huawei.libin@huawei.com
2017/06/20

HUAWEI TECHNOLOGIES CO., LTD.
Agenda

- Livepatch overview
- Livepatch on x86
- Obstacle for livepatch on ARM64
- Solution to support livepatch on ARM64
- Upstream status for ARM64 support
- Questions?
Livepatch overview

- Linux kernel dynamic live patching framework
  - Patch a running kernel
  - No reboots, no disruption to applications
  - Used for security and stability fixes
  - Oracle’s Ksplice / Suse’s Kgraft / Redhat’s Kpatch
- Nov. 2014, Seth Jennings (Red Hat) submitted the first version to the mailing list
- Feb. 2015, been merged in upstream (Linux 4.0)
- By now, only supports x86/s390/powerpc architecture
Livepatch overview

Before Patching:

- Call: old function
- Return: nop

After Patching:

- Call: old function
- Call mcount
- Call ftrace
- Call livepatch
- Return: new function
- Return: nop

Before Patching:

- Call: old function
- Return: nop

After Patching:

- Call: old function
- Call mcount
- Call ftrace
- Call livepatch
- Return: new function
- Return: nop
Livepatch overview

- Livepatch based on ftrace (FTRACE_WITH_REGS)
- FTRACE_WITH_REGS based on gcc profile-before-prologue feature (mfentry)
- But gcc for some arch (including ARM64), NOT support this feature
- So how to remove the obstacles for these arch such as ARM64 …..
Livepatch overview

- Livepatch based on ftrace
- Ftrace based on gcc profile feature
- What’s the gcc profile feature
  - use gcc –pg option to generate profile instruction
Livepatch overview

- Based on general gcc profile:
Livepatch overview

- Limitations:
  - Old func & new func must have the same prologue
  - Otherwise, the control flow will be wrong!
Livepatch on X86

- The solution to the limitation:
  - Gcc’s profile-before-prologue feature (mfentry)
  - Put the profile instruction at the entry of the function
  - Ftrace on x86 based on profile-before-prologue feature now
  - Only some architectures support this feature, but ARM64 not included!
Livepatch on X86

- Based on gcc profile-before-prologue (-mfentry) on X86:

```
The call inst automatically pushes pc to the stack (by hardware)

The call inst automatically pushes pc to the stack (by hardware)

Livepatch ftrace hook

Livepatch ftrace hook

change regs->ip

......

return

return
```

The call inst automatically pushes pc to the stack (by hardware)
Obstacle for livepatch on ARM64

- Why not support profile-before-prologue feature on ARM64?
  - Procedure Call Standard brings the obstacle
  - On X86, the call instruction put the return address into the stack by hardware
  - On ARM64, the call instruction (BL or BLR) transfer the return address into the link register (LR)
  - So the LR should be saved by subroutine (in prologue), that the first routine instruction is never a call instruction on ARM64
Solution to support livepatch on ARM64

- How to solve the obstacle and implement the profile-before-prologue?
- Extend profile instruction into multiple instructions

✔ Save the LR in Stack

```
stp x29, x30, [sp, #16]!
mov x29, sp
bl <mcount>
ldp x29, x30, [sp], #16
<function prologue> >>
...
```

✔ Save the LR in Corruptible Register

```
mov x9, x30
bl <mcount>
mov x30, x9
<function prologue>
...
```

Advantages:

- Smaller impact on performance
- More convenient adaption for ftrace code

```
mov x9, x30
nop <--------->
bl __fentry__
mov x30, x9
<function prologue>
```

Solution to support livepatch on ARM64

- Solution to solve the obstacle on ARM64

```
function foo()
{
    mov x9, x30
    bl mcount
    mov x30, x9
    [prologue]
    [func body]
    return
}

function foo_fix()
{
    mov x9, x30
    bl mcount
    mov x30, x9
    [prologue]
    [func body]
    return
}

function mcount()
{
    save reg
    bl stub
    restore reg
    ret to reg->ip
}

lugroup
{
    change reg->ip
    ...... 
    return
}
```
Upstream status for ARM64 support

- May 2015, I submitted the livepatch support patches for ARM64
  - Add support on ARM64 in kernel (including ftrace adaptation)
    https://lwn.net/Articles/646317/
  - Gcc support on ARM64
    - Support –mfentry feature on ARM64 in gcc
      https://gcc.gnu.org/ml/gcc-patches/2016-03/msg00756.html
Upstream status for ARM64 support

- Gcc support on ARM64
  - Add a new option \texttt{-fpatchable-function-entry=N,M}
    - generate a pad of N nops at beginning of each function suggested by Maxim Kuvyrkov from Linaro firstly
      \url{http://thread.gmane.org/gmane.comp.gcc.devel/139984}
    - Now, Torsten Duwe from Suse with best effort to push this feature
      \url{https://gcc.gnu.org/ml/gcc-patches/2017-05/msg00213.html}
Upstream status for ARM64 support

- Further step after the gcc feature support for ARM64
  - ARM64 Ftrace & Livepatch adaptation
  - Patch module tools adaptation that convert diff patch to a patch module conveniently (such as kpatch-build from Redhat)

https://github.com/dynup/kpatch/tree/master/kpatch-build
Questions?
Thank you

www.huawei.com