

Obstacles & Solutions for Livepatch Support on ARM64 Architecture

www.huawei.com

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

HUAWEI TECHNOLOGIES CO., LTD.



Agenda

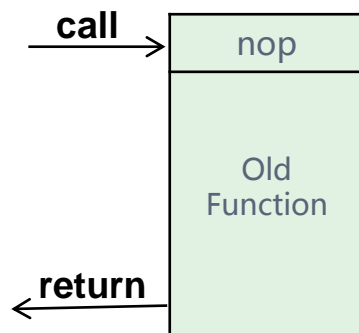
- **Livepatch overview**
- **Livepatch on x86**
- **Obstacle for livepatch on ARM64**
- **Solution to support livepatch on ARM64**
- **Upstream status for ARM64 support**
- **Questions?**

Livepatch overview

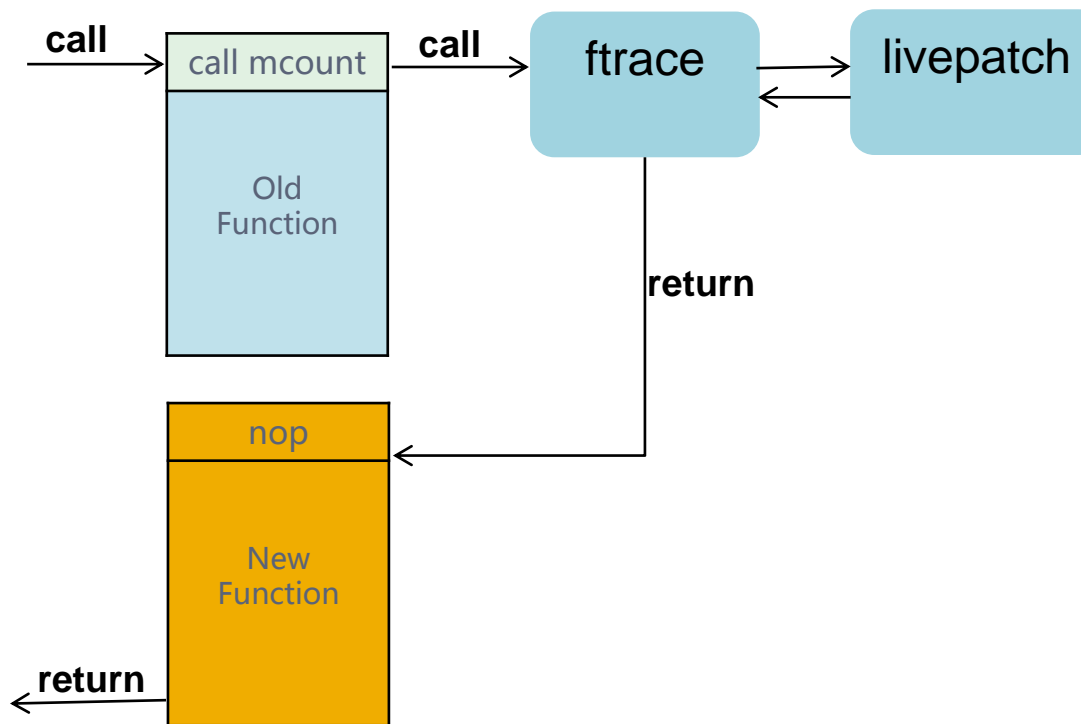
- **Linux kernel dynamic live patching framework**
 - **Patch a running kernel**
 - **No reboots, no disruption to applications**
 - **Used for security and stability fixes**
 - **Oracle's Ksplice / Suse's Kgraft / Redhat's Kpatch**
- **Nov. 2014, Seth Jennings (Red Hat) submitted the first version to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

Livepatch overview

Before Patching:



After Patching:



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

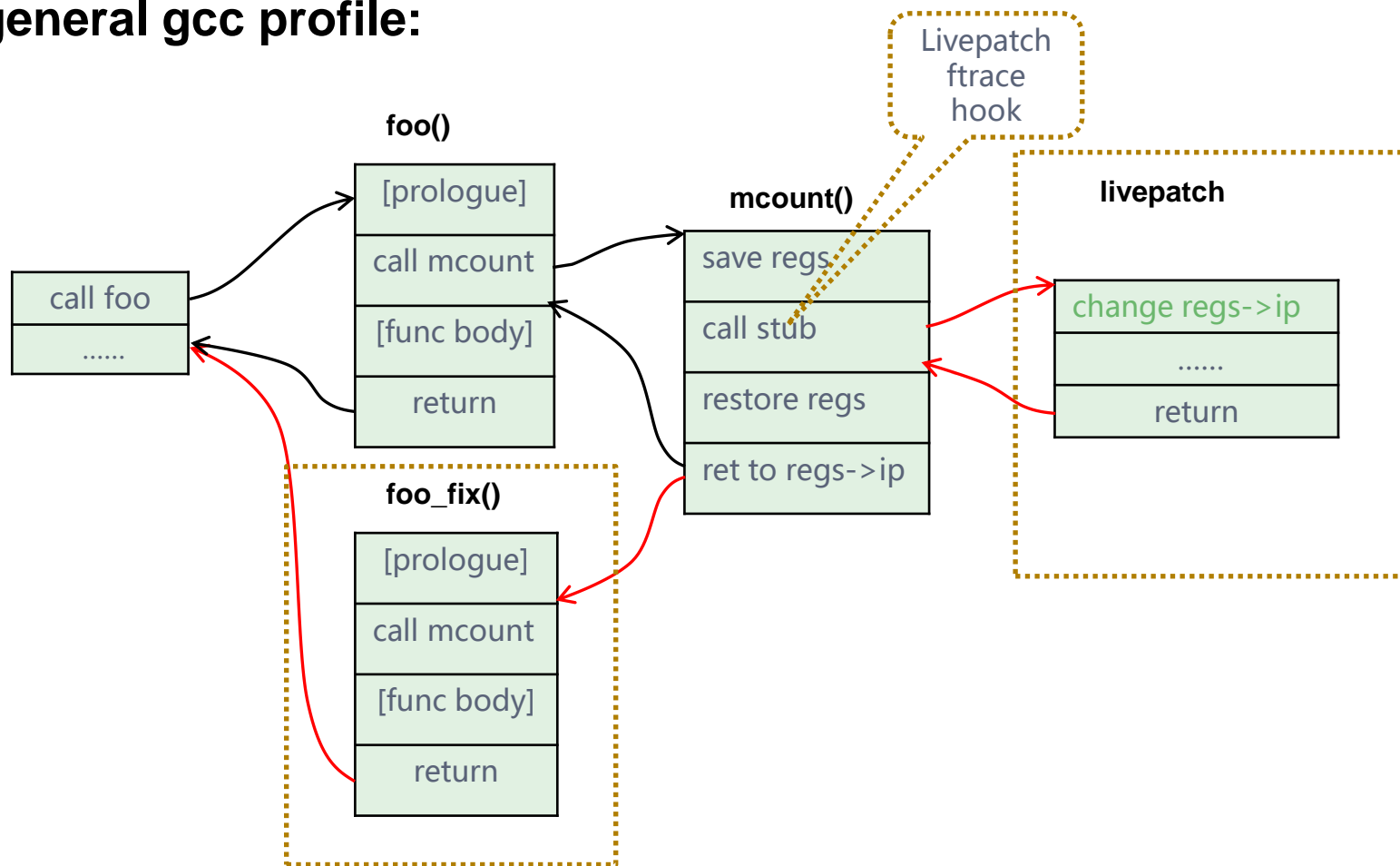
- Livepatch based on ftrace
- Ftrace based on gcc profile feature
- What's the gcc profile feature
 - use gcc `-pg` option to generate profile instruction

```
func:
.LFB0:
    pushq    %rbp
    movq     %rsp, %rbp
    subq     $16, %rsp
    movl     %edi, -4(%rbp)
    movl     $.LC0, %edi
    call     puts
    movl     $0, %eax
    leave
    ret
```

```
func:
.LFB0:
    pushq    %rbp
    movq     %rsp, %rbp
    subq     $16, %rsp
    call     mcount
    movl     %edi, -4(%rbp)
    movl     $.LC0, %edi
    call     puts
    movl     $0, %eax
    leave
    ret
```

Livepatch overview

- Based on general gcc profile:



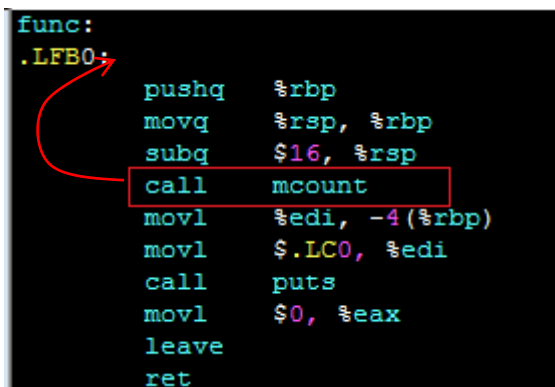
Livepatch overview

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

<pre>#include <stdio.h> int func(int a) { printf("hello\n"); return 0; } ~</pre>			<pre>#include <stdio.h> int func(int a) { int abc = 10; printf("hello\n"); return 0; }</pre>		
test.c	7,1	All	test1.c	1,1	All
<pre>func: .LFB0: pushq %rbp movq %rsp, %rbp subq \$16, %rsp call mcount movl %edi, -4(%rbp) movl \$.LC0, %edi call puts movl \$0, %eax leave ret</pre>			<pre>func: .LFB0: pushq %rbp movq %rsp, %rbp subq \$32, %rsp call mcount movl %edi, -20(%rbp) movl \$10, -4(%rbp) movl \$.LC0, %edi call puts movl \$0, %eax leave ret</pre>		

Livepatch on X86

- The solution to the limitation:
 - Gcc's profile-before-prologue feature (mfentry)

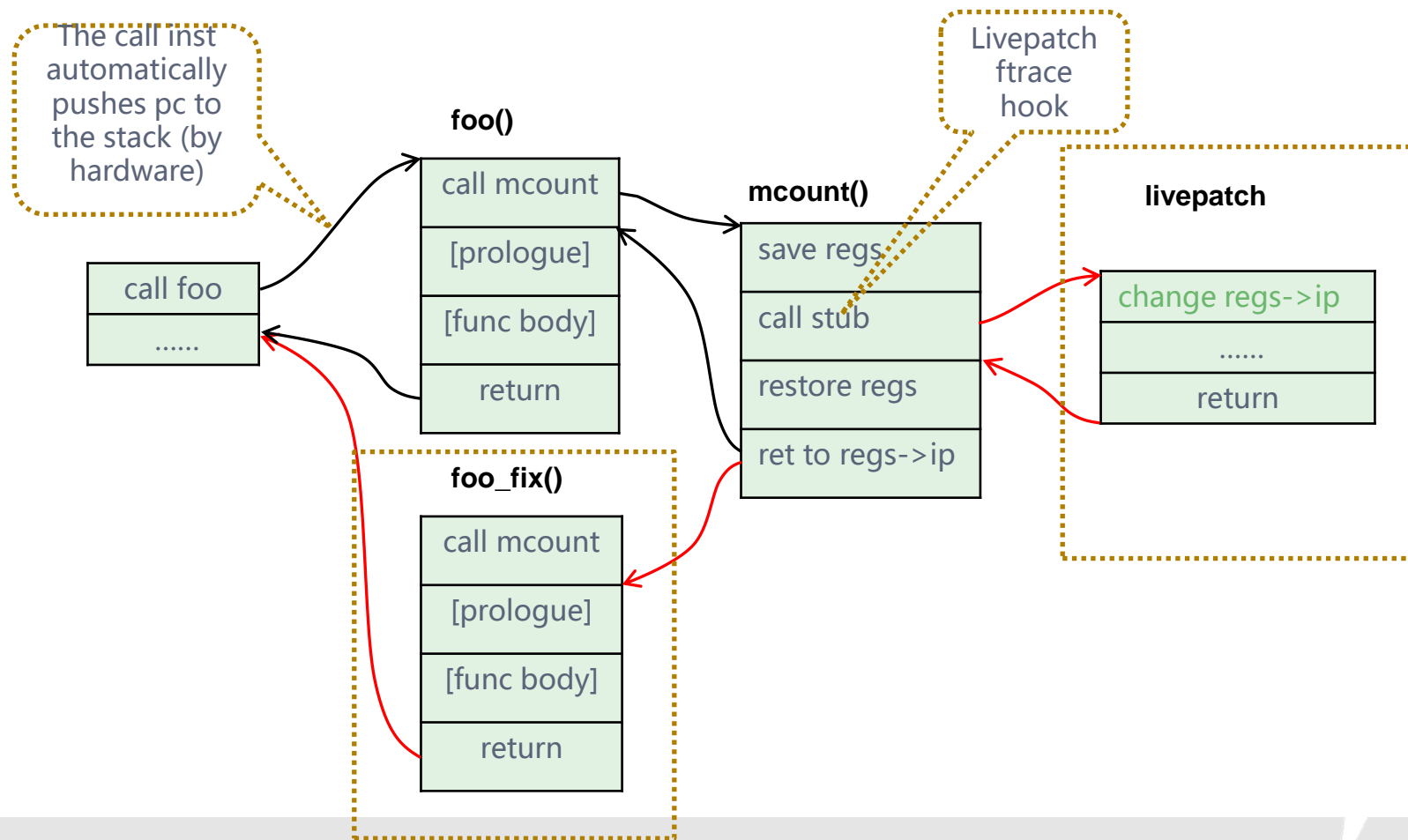


```
func:
.LFB0:
    pushq    %rbp
    movq     %rsp, %rbp
    subq     $16, %rsp
    call     mcount
    movl     %edi, -4(%rbp)
    movl     $.LC0, %edi
    call     puts
    movl     $0, %eax
    leave
    ret
```

- 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:



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:

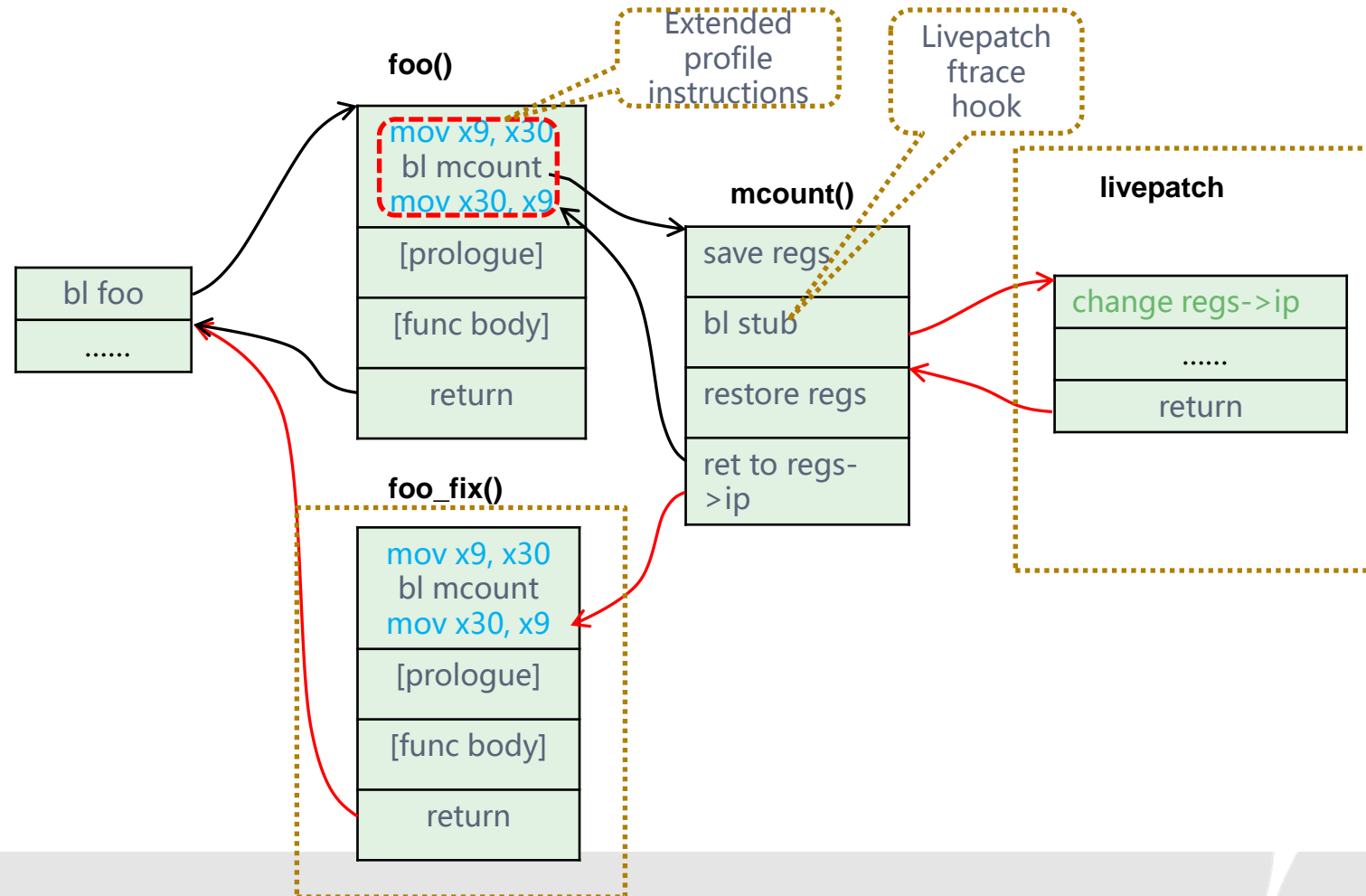
- Smaller impact on performance
- More convenient adaption for ftrace code

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

- **Discussion:** <http://lists.infradead.org/pipermail/linux-arm-kernel/2016-January/401352.html>

Solution to support livepatch on ARM64

- Solution to solve the obstacle on ARM64



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>

[PATCH] [AArch64] support `-mfentry` feature for arm64

```
• From: Li Bin <huawei.libin@huawei.com>
• To: <gcc-patches@gcc.gnu.org>
• Cc: <marcus.dshawcroft@arm.com>, <richard.dearnshaw@arm.com>, <andrew.d.wafaa@infrahead.org>, <takahiro.akashi@linaro.org>, <guochan.jun@huawei.com>, <felix.d@infrahead.org>
• Date: Mon, 14 Mar 2016 16:14:19 +0800
• Subject: [PATCH] [AArch64] support -mfentry feature for arm64
• Authentication-results: sourceware.org; auth=none
```

As ARM64 is entering enterprise world, machines can not be stopped for some critical enterprise production environment, that is, live patch as one of the RAS features is increasing more important for ARM64 arch now.

Now, the mainstream live patch implementation which has been merged in Linux kernel (x86/x86_64) is based on the 'ftrace with regs' feature, and this feature needs the help of gcc.

This patch proposes a generic solution for arm64 gcc which called `mfentry`, following the example of x86, x86_64, etc. and on these archs, this feature has been used to implement the ftrace feature 'ftrace with regs' to support live patch.

By now, there is another solution from linaro [1], which proposes to implement a new option `-fpilogue-pad` that generate a pad of N nops at the beginning of each function. This solution is a arch-independent way for gcc, but there may be some limitations which have not been recognized for Linux kernel to adapt to this solution besides the discussion on [2], typically for powerpc archs. Furthermore I think there are no good reasons to promote the other archs (such as x86) which have implemented the feature 'ftrace with regs' to replace the current method with the new option, which may bring heavily target-dependent code adaption, as a result it becomes a arm64 dedicated solution, leaving kernel with two different forms of implementation.

[1] <https://gcc.gnu.org/ml/gcc/2015-10/msg00090.html>
[2] <http://lists.infrahead.org/pipermail/linux-arm-kernel/2016-January/401854.html>

Jiangziji (1):
[AArch64] support -mfentry feature for arm64



Content
Weekly Edition
Archives
Search
Kernel
Security
Distributions
Events calendar
Unread comments

LWN FAQ
Write for us

Edition
Return to the
Kernel page

livepatch: add support on arm64

```
From: Li Bin <huawei.libin@huawei.com>
To: <rostedt@goodmis.org>, <mingo@kernel.org>,
    <jpoimboe@redhat.com>, <sjenning@redhat.com>,
    <jkosina@suse.cz>, <vojtech@suse.cz>,
    <catalin.marin@arm.com>, <will.deacon@arm.com>,
    <masami.hiramatsu.pt@hitachi.com>
Subject: [RFC PATCH 0/5] livepatch: add support on arm64
Date: Thu, 28 May 2015 13:51:00 +0800
Message-ID: <1432792265-24076-1-git-send-email-huawei.libin@huawei.com>
Cc: <live-patching@vger.kernel.org>, <linux-arm-kernel@lists.infradead.org>, <linux-kernel@vger.kernel.org>,
    <lizefan@huawei.com>, <felix.yang@huawei.com>,
    <guochan.jun@huawei.com>, <xiexiuxi@huawei.com>,
    <huawei.libin@huawei.com>
```

Archive-link: [Article](#)

This patchset propose a method for gcc `-mfentry` feature(profile before prologue) implementation for arm64, and propose the livepatch implementation for arm64 based on this feature.

The gcc implementation about this feature will be post to the gcc community soon.

With this `-mfentry` feature, the entry of each function like:

```
foo:
    mov x9, x30
    bl __fentry__
    mov x30, x9
    [prologue]
    ...
```

The x9 is a callee corruptible register, and the `__fentry__` function is responsible to protect all registers, so it can be used to protect the x30. And the added two instructions which is register mov operation have relatively small impact on performance.

This patchset has been tested on arm64 platform.

Li Bin (4):
livepatch: ftrace: arm64: Add support for DYNAMIC_FTRACE_WITH_REGS
livepatch: ftrace: add ftrace_function_stub_ip function
livepatch: ftrace: arm64: Add support for -mfentry on arm64
livepatch: arm64: add support for livepatch on arm64

Upstream status for ARM64 support

- **Gcc support on ARM64**
 - ▣ **Add a new option -fpatchable-function-entry=N,M**
 - generate a pad of N nops at beginning of each function suggested by Maxim Kuvyrkov from Linaro firstly
<http://thread.gmane.org/gmane.comp.gcc.devel/139984>
 - Now, Torsten Duwe from Suse with best effort to push this feature
<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

Copyright©2009 Huawei Technologies Co., Ltd. All Rights Reserved.