LLVM平行宇宙计划及openEuler Embedded场景实践分享

openEuler TC委员　Compiler SIG Maintainer

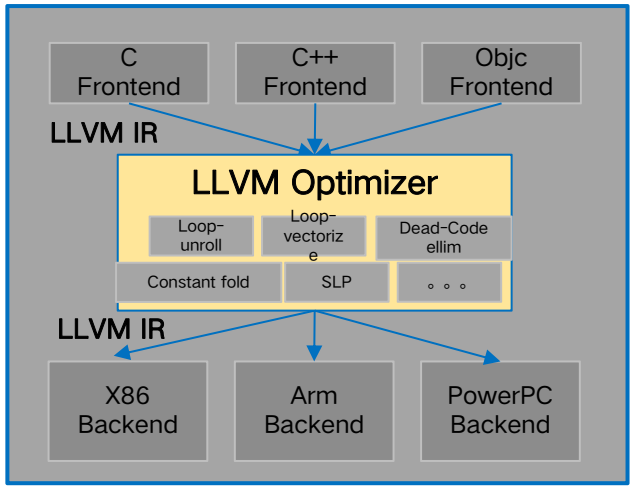
赵川峰

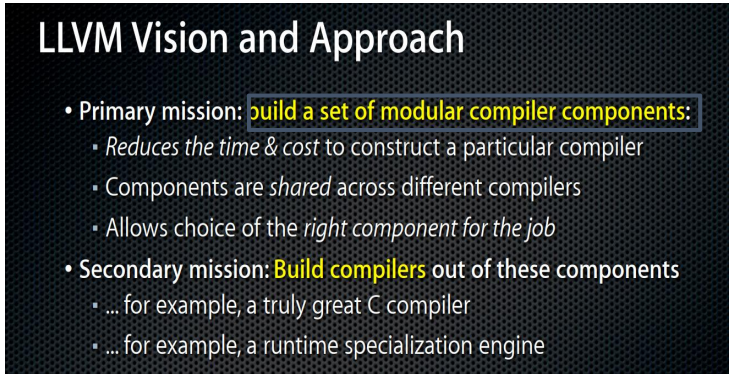
1. **为什么是Clang+LLVM**
2. LLVM平行宇宙计划简介
3. Embendded场景实践
4. 服务器场景进展

LLVM架构良好、License友好、社区繁荣活跃

- **架构良好：**模块化解耦，统一的IR表示，强大的Pass系统

- **协议友好：**LLVM v9.0后属于Apache License，License对各个大厂来说更加友好

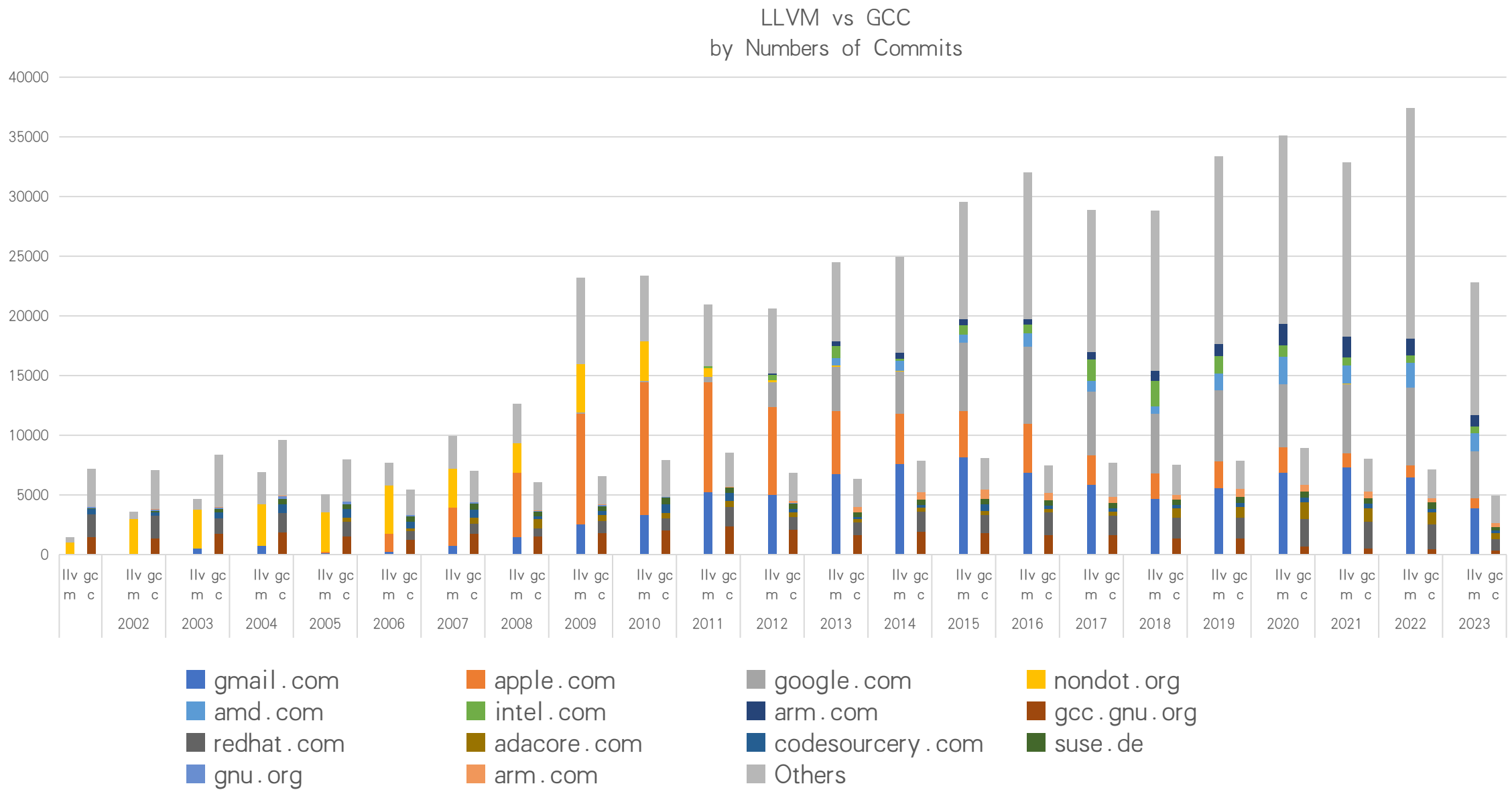




摘自《Introduction to the LLVM Compiler System -- Chris Lattner 》

- **繁荣的生态活跃度：**截至目前，LLVM社区社区贡献者已经达到2634人，2022年增加340人，涉及公司150+，周Commit数量超500+以上。

LLVM vs GCC社区参与者与活跃度



关键价值点

OpenEuler

- **提升代码质量**
  - 语言标准遵从性能好，提供丰富、强大的sanitizer能力，具备更全面的安全保护特性;
- **性能潜力大**
  - 架构友好，更容易实现优化;
- **利于技术创新**
  - MLIR、CIRCT、BOLT等新编译能力/特性不断涌现LLVM论文/顶尖学术分享数量大幅领先GCC;
- **人才获取**
  - 大学生参与LLVM社区远大于GCC，Top企业商用编译器转型LLVM，专业人才可获得性高;

业界Clang/LLVM build Linux的进展

- 目前Clang+LLVM和GCC是Linux kernel官方支持的两款编译器。
- LLVM构建openEuler 23.09kernel通过。

The following architectures are targetable from both LLVM and the Linux kernel and are relatively well supported and tested:

- arm
- arm64
- hexagon
- s390
- x86

The following architectures are targetable from both LLVM and the Linux kernel and are known to work in specific/limited configurations and supported on a best-effort basis:

- mips
- powerpc
- riscv
- um (usermode)

- Android、ChromeOS、OpenMandriva、Chimera Linux使用Clang构建的kernel发布distributions；
- Google和Meta在其数据中心运行Clang构建的kernel；

Building Linux with Clang/LLVM

This document covers how to build the Linux kernel with Clang and LLVM utilities.

About

The Linux kernel has always traditionally been compiled with GNU toolchains such as GCC and binutils. Ongoing work has allowed for Clang and LLVM utilities to be used as viable substitutes. Distributions such as Android, ChromeOS, OpenMandriva, and Chimera Linux use Clang built kernels. Google's and Meta's datacenter fleets also run kernels built with Clang.

- Ubuntu/Fedora同时支持LLVM与GCC编译器。

1. 为什么是Clang+LLVM
2. **LLVM平行宇宙计划简介**
3. Embendded场景实践
4. 服务器场景进展

什么是LLVM平行宇宙计划

- 起因
  - 目前openEuler社区默认编译器为GCC，如何平滑地过渡到LLVM？
- 目的
  - 提升社区代码质量、代码安全；
  - 提升openEuler场景化竞争力；
- LLVM平行宇宙计划
  - 使能LLVM编译器构建更多的openEuler软件包，挑战基于LLVM技术栈完成openEuler版本发布，这个工作是平行与目前openEuler版本发布工作的，所以称为“LLVM平行宇宙计划”。
- oEEP
  - CompielIr SIG提交《LLVM平行宇宙计划--基于LLVM技术栈构建oE软件包》

1. 为什么是Clang+LLVM
2. LLVM平行宇宙计划简介
3. **Embendded场景实践**
4. 服务器场景进展

Embendded场景进展

目前进展： ARM64平台已支持Clang/LLVM构建版本

https://openeuler.gitee.io/yocto-meta-openeuler/master/features/clang_llvm.html

meta-clang 层介绍 ¶

`meta-clang` 层包含使用 clang/llvm 编译器工具链所需要的recipes和bbclass文件，由于 openeuler 使用 external-toolchain 机制，无需在 yocto 工程中编译 clang/llvm 工具链。

meta-clang层中主要起作用的是clang.bbclass文件，该文件用来控制编译时传入clang/llvm编译器工具链变量和依赖，除此之外，目前还有一小部分软件包并不完美支持使用clang/llvm来编译，nonclangable.conf文件记录了这些软件包的情况。

codesize：

- Kernel： llvm优于gcc 1.29%
- 其他软件包codesize总和：llvm优于gcc1.9%
  - 544个可执行文件，263个文件size，llvm优于gcc，248持平
  - 282个so文件，84个文件size，llvm优于gcc，157持平

编译时间：

- gcc： 2483-tasks 耗时 2033.05s
- llvm：2505-tasks 耗时 1751.05s
- 约 16%的速度提升

性能测试：

- Image用gcc构建，coremark分别用clang，gcc编译，性能提升6%；
- Image和coremark分别用clang，gcc编译构建，性能提升6%；

Embendded场景软件包修复

共154个软件包，一次性通过143个，修改编译选项通过5个，修改应用源码通过3个，使用gcc编译2个

软件包	错误信息	修改方法	修改方法详情
dhcp	error: expected parameter declarator extern int asprintf(char **strp, const char *fmt, ...);	修改源码	clang 不支持 glibc 中使用的 va_arg_pack 相关函数，导致用户代码对该函数的声明（stdio2.h）与编译器头文件（missing.h）的声明有冲突。解决方案是通过在用户代码中添加宏屏蔽相关声明。
openamp	error: member reference base type 'atomic_int' (aka '_Atomic(int)') is not a structure or union	修改源码	问题定位为用户代码对 atomic_flag 相关函数的使用不符合C11标准，传入了不匹配的参数类型，gcc的实现相对宽松未报错。解决方案是修改用户代码，将有问题的参数类型替换掉。
busybox	运行时segmentation fault	修改源码	应用代码有未定义行为，修改源码
gnutls	clang-15: error: unsupported argument 'all' to option '-march='	修改选项	clang 不支持"-Wa,-march=all"，删除'AM_CCASFLAGS = -Wa,-march=all'。
libmetal	error: macro 'ATOMIC_VAR_INIT' has been marked as deprecated [-Werror,-Wdeprecated-pragma]	修改选项	ATOMIC_VAR_INIT已经被废弃，因为不适配所有场景。#pragma clang 会提示废弃warning。 但是当前工程的编译选项中存在-Werror，将告警转为error。解决方案为删除-Werror，或者增加-Wno-error=deprecated-pragma 。
grub	build-grub-module-verifier: error: search_label: unsupported relocation 0x108	使用GCC	llvm存在‘PIC’+‘mcmode=large’产生错误重定位问题，未解决，nonclangable中原生规避
pseudo	./ports/linux/pseudo_wrappers.c:80:14: error: use of unknown builtin 'builtin_apply' [-Wimplicit-function-declaration]	使用GCC	Host侧软件，暂用GCC构建
dsoftbus	error: unknown warning option '-Wno-maybe-uninitialized'; did you mean '-Wno-uninitialized'? [-Werror,-Wunknown-warning-option]	修改选项	添加patch修改选项指定toolchain meta-openeuler/dynamic-layers/clang-layer/recipes-core/dsoftbus/dsoftbus/0001-change-toolchain-for-clang-build.patch
lxc	error: unknown warning option '-Werror=stringop-overflow'; did you mean '-Werror=shift-overflow'? [-Werror,-Wunknown-warning-option]	修改选项	添加-Wno-error=unused-command-line-argument
cracklib	编译warning报error	修改选项	添加 -Wno-int-conversion

https://github.com/OpenAMP/open-amp/pull/472

Embendded场景软件包修复——举例

OpenEuler

Busybox运行失败

```c
#include <stdlib.h>

struct AA {
 int a;
 int b;
};

struct AA *const p_g;

static void* not_const_pp(const void *p) { return (void*)p; }
#define barrier() __asm__ __volatile__("":::"memory")

void test()  {
  (*(struct AA**)not_const_pp(&p_g)) = malloc(sizeof(*p_g));
  barrier();
  (*p_g).a = -1;
  (*p_g).b = 10;
}

int main() {
  test();
  return 0;
}
```

问题分析：https://gitee.com/src-openeuler/busybox/issues/I64UUK?from=project-issue

下一步工作思路

- **Codesize优化**
  - 使能HyBF分支融合；
- **LLVM sanitizer支持**
  - 支持compiler-rt，libunwind，使能ASAN、HWASAN、HCFI等特性
- **RUST语言支持**
  - 支持RUST语言
  - 增强C/C++/RUST混编能力

1. 为什么是Clang+LLVM
2. LLVM平行宇宙计划简介
3. Embendded场景实践
4. **服务器场景进展**

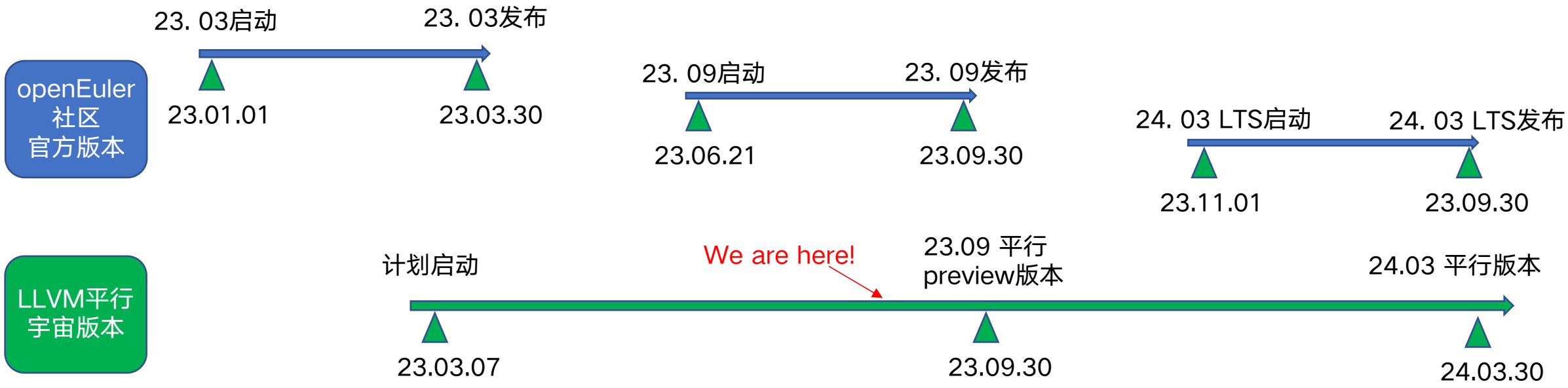
LLVM平行宇宙计划目前进展

- **运作**：LLVM平行宇宙计划运作
  - 基于软件所OBS构建环境，以openEuler 23.03版本为基线，默认切换Clang+LLVM为构建编译器，完成构建基线；
  - 软件所、华为、SUSE、统信及爱好者参与，建立微信群，文档贡献目录，双周周例行工作会议；

	软件包名	负责人	协助人	状态	领取时间	根因分类
235	pacemaker	孙越池		修复完成	2023年5月5日	
236	systemtap	孙越池		需要帮助	2023年5月5日	
237	bcc	章翔		需要帮助	2023年5月5日	
238	brltty	章翔		进行中	2023年5月5日	
24	22 zip	黄筱雅		修复完成	2023年4月11日	makefile写死
25	23 libssh	Chenxi Mao		修复完成		源码问题
26	24 libaio	Chenxi Mao		修复完成		源码问题
27	25 python3	Chenxi Mao		进行中		
28	26 libvirt	Chenxi Mao		修复完成		源码问题
29	27 audit	林敬淞		修复完成	2023年4月16日	源码问题
30	28 busybox	林敬淞		修复完成	2023年4月16日	spec写死
508	libcareplus	黄筱雅	孙越池	需要帮助	2023年6月19日	makefile写死 werror问题 环境问题
	libmetal	孙越池	章翔	修复完成	2023年6月19日	
	dde-device-formatter	李林杰		修复完成		

进展：北向兼容性—工作策略及计划

- 跟随社区节奏，通过小的兴趣小组发布平行版本，验证LLVM构建openEuler可行性；



23.03平行版本OBS工程：https://build.tarsier-infra.com/project/show/Mega:23.03
23.09平行版本OBS工程：https://build.tarsier-infra.com/project/show/Mega:23.09

进展：北向兼容性—软件包修复进展

- **进展：**LLVM平行宇宙计划当前进展

架构	类别	软件包个数	初始失败个数	剩余失败个数
Aarch64	全量OS	4307	439	69
	EPOL	1107	56	28
x86_64	全量OS	4305	820	230
	EPOL	1107	137	89
riscv64	全量OS	4305	459	76
	EPOL	1107	54	25

已修复问题错误类型分析

错误类型	占比	修改方式
环境变量设置问题	13%	修改spec文件
Spec脚本写死GCC	20%	修改spec文件
Makefile等构建脚本写死GCC	28%	修改构建脚本
源码编写不符合语言规范	15%	修改源代码
LLVM报错机制严格	11%	修改源代码&压制错误
LLVM不兼容GCC的选项	6%	编译器增加选项支持
LLVM dwarf格式不兼容	2%	修改编译器
LLVM编译器缺陷	5%	修改编译器

未修复问题错误类型分析

错误类型	占比	修改方式
环境变量设置问题	11%	修改spec文件
Spec脚本写死GCC	17%	修改spec文件
Makefile等构建脚本写死GCC	10%	修改构建脚本
源码编写不符合语言规范	8%	修改源代码
LLVM报错机制严格	12%	修改源代码&压制错误
LLVM不兼容GCC的选项	2%	编译器增加选项支持
Gfortran缺失	12%	提供flang编译器
测试用例失败/待分析	12%	分析失败用例

- 关键软件包的现状

依据QA SIG发布的社区软件包质量分级策略，L1、L2软件共80个软件包。

- 除glibc外，其他软件包均已构建通过；
- glibc 2.34修改后，验证通过，2.38待验证；

https://docs.qq.com/sheet/DTGlnVUFESUVXWEFE?tab=BB08J2

总体看来，编译器本身问题占比小，绝大部分修改环境变量和源代码

进展：南向兼容性

OpenEuler

- 问题：kernel用LLVM构建，模块/驱动用GCC构建，是否存在问题？

符号检查

gcc & clang 编译kernel-2303(6.1.8)差异表

no	gcc编译相比clang增加的符号	clang编译相比gcc变化的符号
1	*_attribute_ ((_externally_visible_))*	
2	*_attribute_ ((_noclone_))*	
3	*_attribute_ ((_designated_init_))*	
4	*_attribute_ ((no_sanitize_address))*	
5	s#poll_table_entry inline_entries [((**832** - 256) / sizeof (struct poll_table_entry))]	s#poll_table_entry inline_entries [((**768** - 256) / sizeof (struct poll_table_entry))]

数据对齐

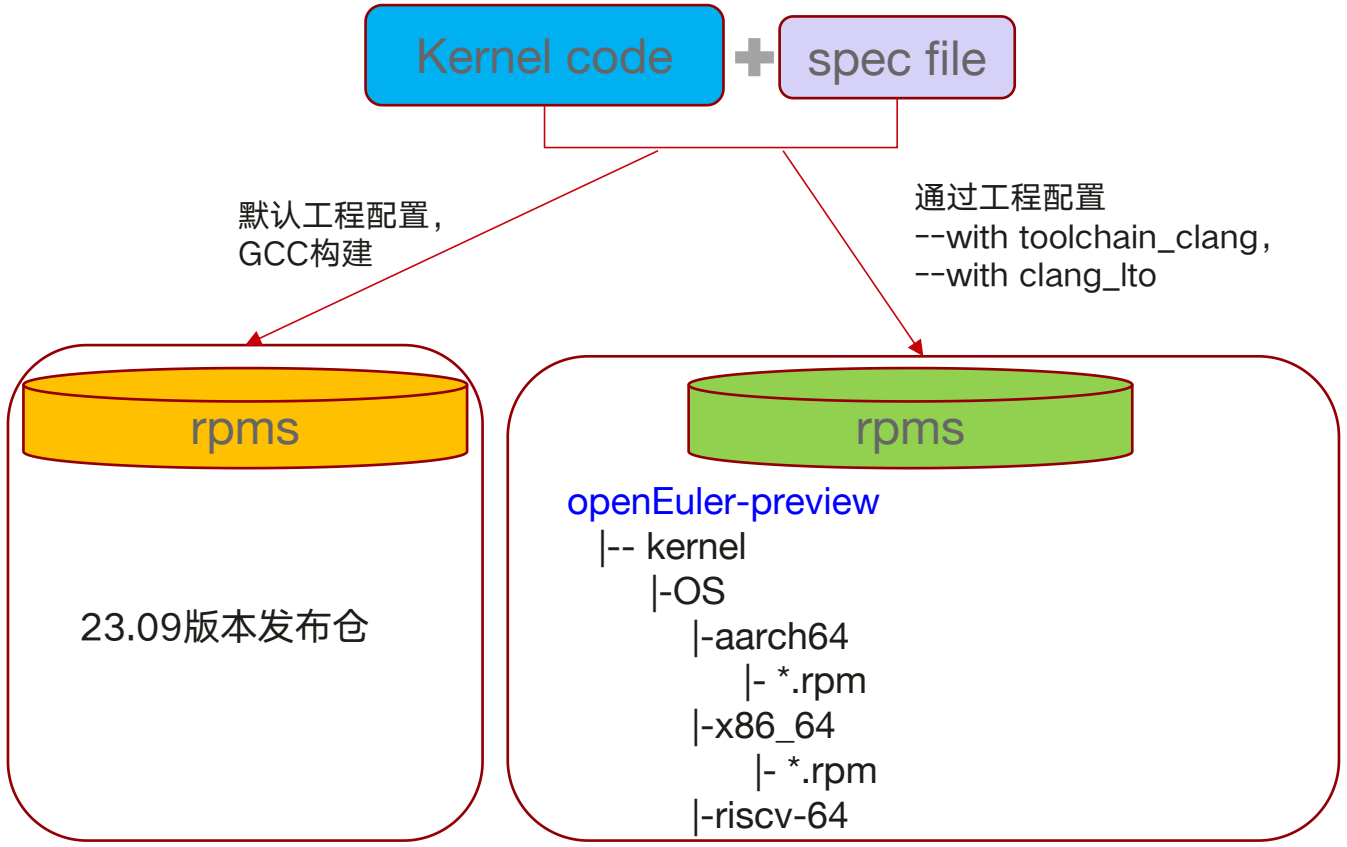
```
In sg_set_buf():

unsigned int offset = (unsigned long) buf & ~(~((1UL << 12) - 1));  /* 0x0FFF */
```

```
static int cts_cbc_decrypt(...) {
  u8 s[bsize * 2], d[bsize * 2];
    ...
  sg_set_buf(&sgsrc[0], s + bsize, bsize);
  sg_set_buf(&sgdst[0], d, bsize);  /* 0x0FF0 after inlining */
    ...
```

所以，最好用同一个编译器编译内核态代码~

支持LLVM构建内核在openEuler-preview发布：进行中



方案关键点：

- **代码形式：**同源，极大减少维护量。
- **使用方式：**默认GCC构建，手动拉取LLVM构建版本。

优点：

- **代码质量。**双编译器保证代码质量，减少代码未定义行为。
- **Kernel先行。**LLVM平行宇宙计划中kernel包先做平行版本，**有利于：**驱动厂商发布相关驱动；云和虚拟化场景竞争力验证；

测试与维护：

- **社区测试。**(1) 门禁；（2）kernel sig测试（3）QA sig kernel测试；
- **兼容性测试：**与兼容性SIG确定南向兼容性列表。
- **维护：**Compiler SIG负责LLVM构建相关的维护工作。


OpenEuler

CFM–CS&SEME–Fusion

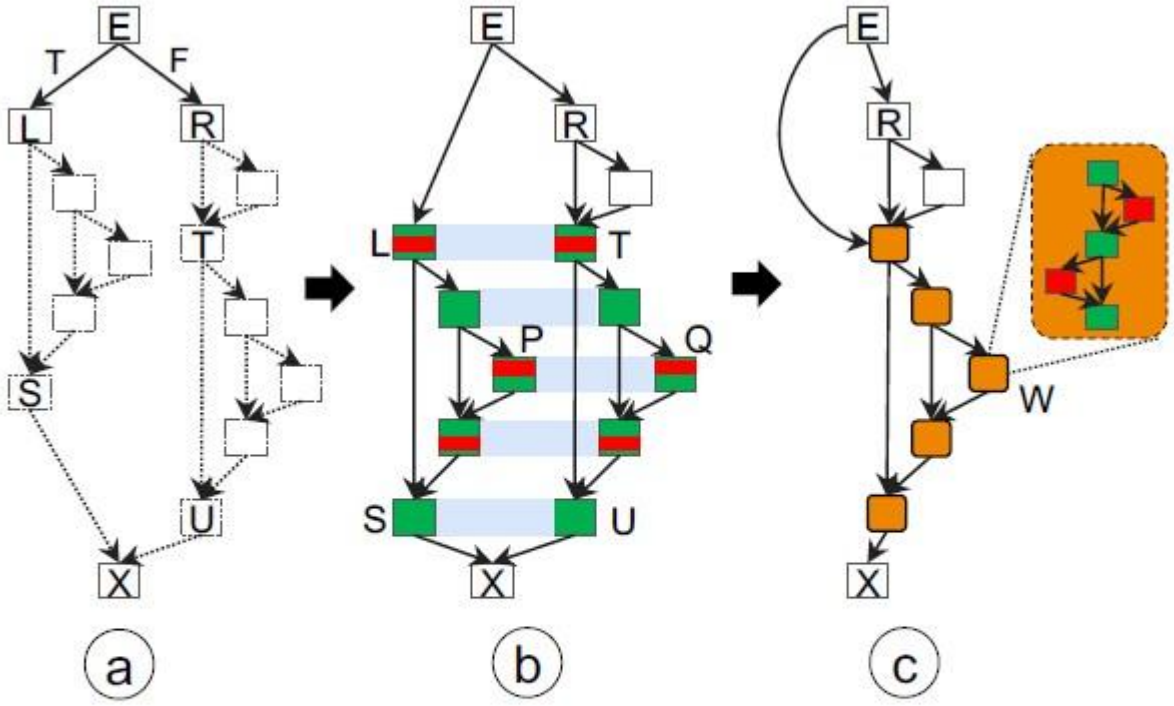


Figure 6. CFM-CS overview. (a) Given an *if-then-else* statement, (b) we identify isomorphic control-flow in the two regions, and (c) we align and merge the corresponding blocks.

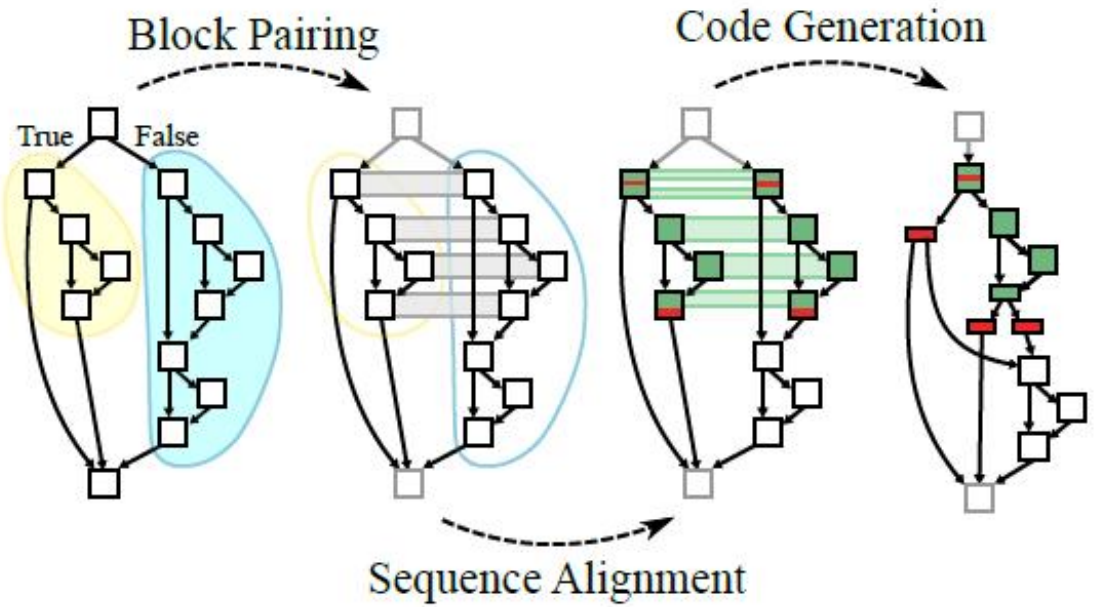


Figure 8. Overview of the SEME-Fusion technique.