openEuler编译器介绍及最新进展

GCC for openEuler 优化介绍及新特性前瞻

LLVM for openEuler 进展介绍及未来规划

开放原子开源基金会 OPENATOM FOUNDATION | OpenEuler

目录

GCC for openEuler 优化介绍及新特性前瞻

GCC for openEuler：使能多样算力，聚焦便捷易用、生态兼容和场景化性能增强

嵌入式 | HPC/AI 高性能 | 安全 | 虚拟化 | ARM原生 | Web | 数据库 | 分布式存储 | 数据中心

GCC for openEuler

编译优化：矢量优化、循环优化

芯片使能：指令优化、智能分配

插件框架：通用插件接口、多编译器支持

CFGO：多模态全流程的反馈优化

基础优化能力

源代码：寄存器优化、死代码消除

二进制：基本块重排、函数重排、内联优化

OS全栈优化：内核反馈优化、库反馈优化、应用反馈优化

A-FOT工具链

openEuler协同：面向场景性能工程、安全/可靠/运维使能

X86系 | ARM系 | LoongArch | RISC-V | 其他国产芯片

- SPEC2017 INT性能**超开源GCC 20%**
- 使能多样算力，**HPC内核函数性能提升30%**

- 一键启用反馈优化，数据库**性能提升18%**
- 一套插件兼容不同编译框架，**使能多样算力差异化编译诉求**

GCC for openEuler使能多样算力，聚焦于便捷易用、生态兼容和场景化性能增强，并在以下四个方向实现主要突破。

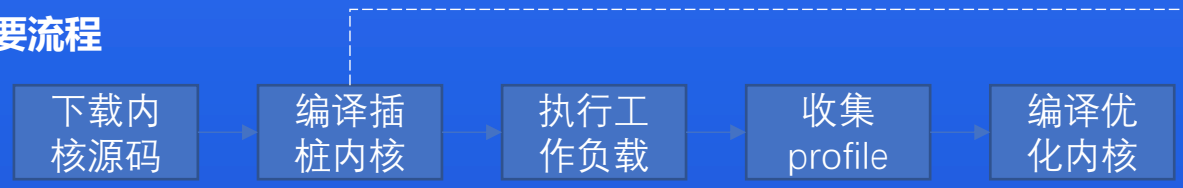
- **基础性能**：基于GCC开源版本构建竞争力，通过在openEuler社区推进鲲鹏特性的支持和短板的补齐，提升通用场景性能。
- **反馈优化**：整合业界领先的反馈优化技术，实现程序全流程和多模态反馈优化，提升数据库等云原生场景重点应用性能。
- **芯片使能**：使能多样算力指令集，围绕内存等硬件系统，发挥算力优势，提升HPC等场景化性能。
- **插件框架**：使能多样算力差异化编译诉求，一套插件兼容不同编译框架，打通GCC和LLVM生态。

主要特性介绍

内核反馈优化

- 通用场景化高性能内核优化方法
- 编译器软件插桩反馈优化（aka PGO、FDO）在openEuler kernel上使能

主要流程



Configure openEuler kernel with

- **CONFIG_PGO_KERNEL=y**
- CONFIG_GCOV_KERNEL=y
- CONFIG_ARCH_HAS_GCOV_PROFILE_ALL=y
- CONFIG_GCOV_PROFILE_ALL=y
- CONFIG_DEBUG_FS=y

适用场景

- 工作负载相对固定
- 期望最大化性能
- 接受重新编译内核

易用性提升

A-FOT新模式：Kernel PGO

使用方式：用户仅需填写指定配置文件，即可通过使用A-FOT一键自动完成整个优化流程，得到新的优化后内核

优化效果

应用	内核反馈优化	内核+应用反馈优化
MySQL	+2~4%	+10~20%
Ceph	+5~7%	+7~10%
Nginx	+5~15%	+10~20%

主要特性介绍

插件框架

- **个性化编译的便捷开发**
- 无需深入修改编译器内部逻辑，帮助开发者可以**实现独立编译优化和编译工具的便捷开发**

- **特性进展**

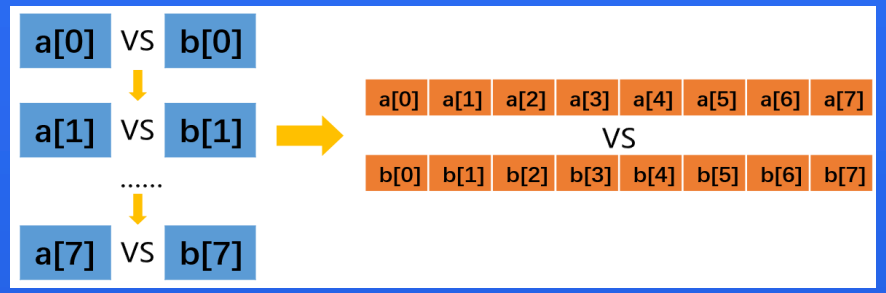
① 能够覆盖80%的GIMPLE
② 支持LTO阶段使能插件
③ LLVM客户端原型
（https://gitee.com/openeuler/pin-llvm-client）

- **欧洲开源峰会（OSSEU23）主题报告**

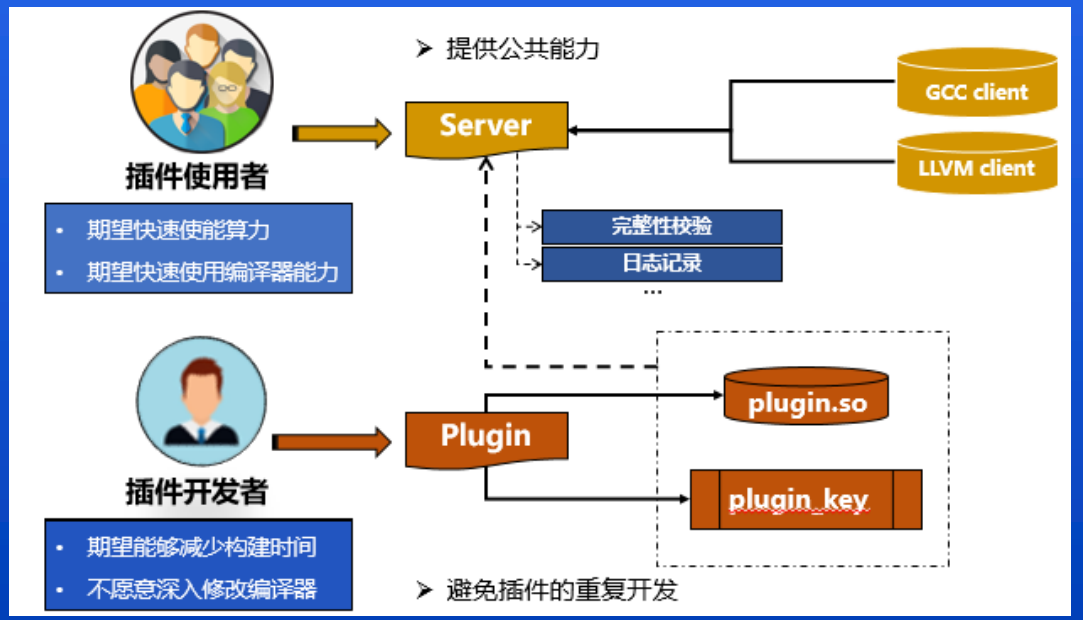
The compiler plugin framework to facilitate customized compilation and development

查询元素扩展优化插件

- 可以支撑GCC for openEuler的优化特性，往插件框架迁移

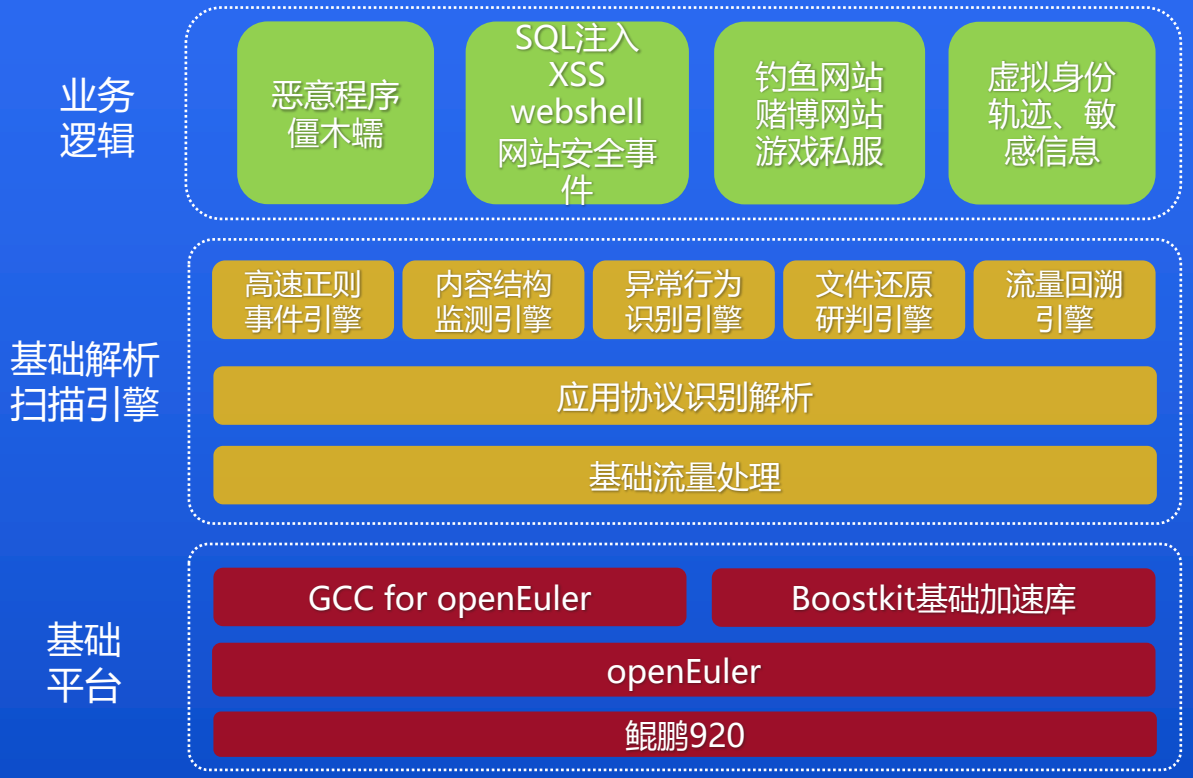


- **效率：** 开发效率由1人月降至**7人天**



客户效果

恒安嘉新：GCC for openEuler加速核心业务性能



恒安嘉新是一家数字应用智能化服务商，深耕网络安全领域，构建面向数字中国基础设施的数字安全保护屏障，包含金川（网络空间治理）、金御（大数据安全应用）、云河（SaaS服务）三大核心业务，与三大运营商合作，服务范围覆盖31省

业务挑战

- **大流量**：单台设备吞吐量大于100Gbps，并发连接数不少于1000万，新建连接不低于每秒5万；
- **高准确**：处置成功应不低于95%，协议识别准确率不低于90%

解决方案

- **鲲鹏亲和优化**：GCC for openEuler通过亲和鲲鹏920的流水线优化，ccmp等指令级优化，实现汇编指令鲲鹏亲和，核心业务性能提升5%
- **全局优化**：GCC for openEuler通过全局优化，实现跨文件函数内联，减小调用开销，跨文件函数常量传播，消除冗余代码，核心业务性能提升5%

客户收益

- **问题定位**：联合GCC for openEuler团队快速定位解决CPU使用率不均衡、系统升级后业务系统严重丢包问题，保障客户DPI&协议解析平台业务稳定运行
- **迁移结果**：GCC for openEuler在恒安嘉新DPI&协议解析场景落地，支撑XXX节点稳定运行，业务性能提升5%~10%

新特性前瞻

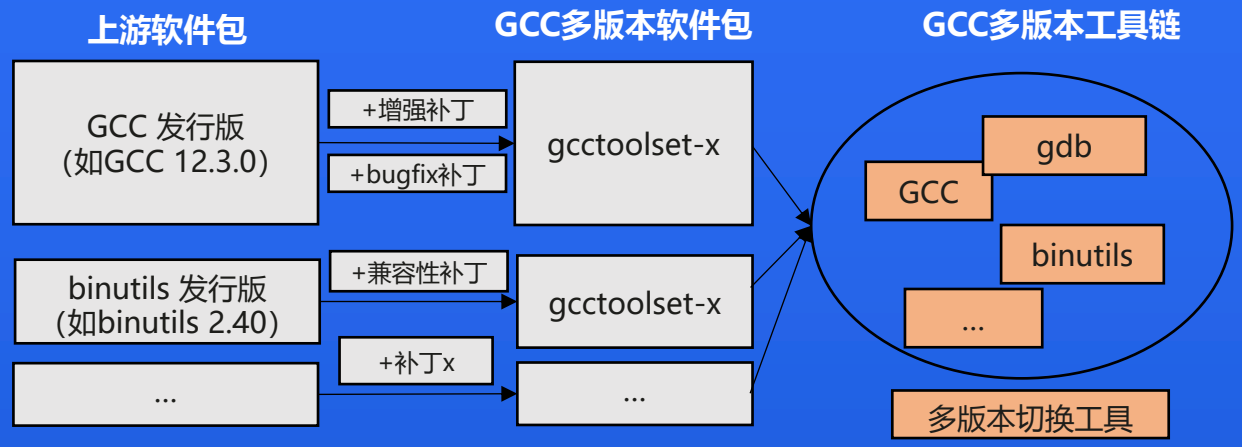
GCC多版本支持

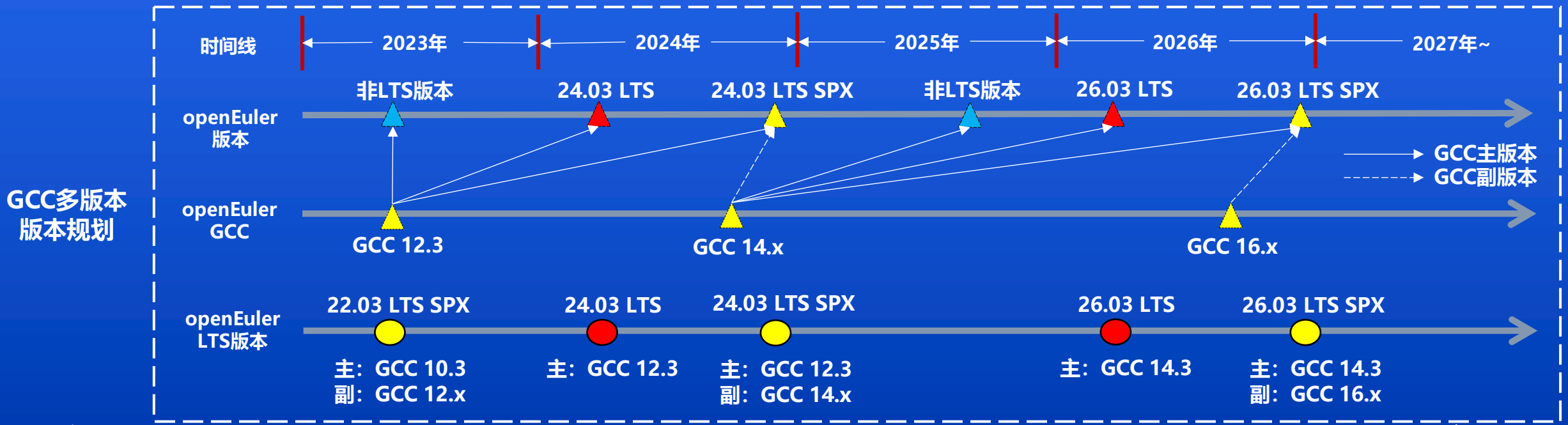
openEuler GCC多版本工具链由三部分组成：

① **上游软件包**：GCC、binutils等上游社区发布件

② **补丁**：bugfix、兼容性等补丁

③ **多版本切换工具**：提供切换工具用于使能工具链





新特性前瞻

基础性能：编译优化能力持续增强

Double-sized mul

识别64位乘法软件算法，转换为指令实现

openssl提升20%

```
uint128_t mul128_perm (uint64_t a, uint64_t b)
{
  uint64_t a_lo = a & 0xFFFFFFFF;
  uint64_t b_lo = b & 0xFFFFFFFF;
  uint64_t a_hi = a >> 32;
  uint64_t b_hi = b >> 32;
  uint64_t lolo = a_lo * b_lo;
  uint64_t lohi = a_lo * b_hi;
  uint64_t hilo = a_hi * b_lo;
  uint64_t hihi = a_hi * b_hi;
  uint64_t middle = hilo + lohi;
  uint64_t middle_hi = middle >> 32;
  uint64_t middle_lo = middle << 32;
  uint64_t res_lo = lolo + middle_lo;
  uint64_t res_hi = hihi + middle_hi;
  res_hi = res_lo < middle_lo ? res_hi + 1 : res_hi;
  res_hi = middle < hilo ? res_hi + 0x100000000 : res_hi;
  uint128_t res = ((uint128_t) res_hi) << 64;
  res += res_lo;
  return res;
}
```

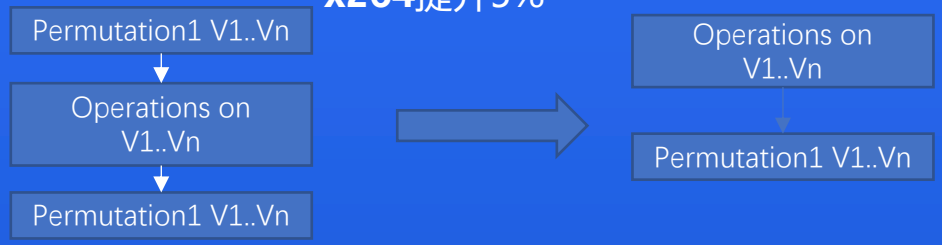
```
mov x2, x1
umulh x1, x0, x1
mul x0, x0, x2
```

利用硬件资源

Forward propagation of permutations

优化矢量化指令数

x264提升5%



优化指令数量

Split complex instructions

针对ldp/stp表现较差的场景，拆分为2个ldr/str

rapidjson提升7%

补充硬件能力

除此之外，还计划推出如下增强优化，期待大家的尝试与反馈：

- LLC Allocation optimization
- CRC32 optimization
- Indirect Call Promotion
- IPA prefetch
- ...

新特性前瞻

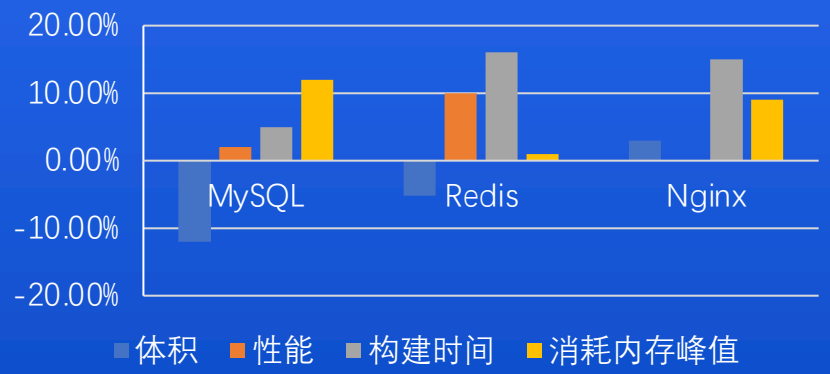
高级优化：LTO和AI4Compiler

LTO（link time optimization） openEuler by default

策略

- 与构建工程团队合作，默认所有C/C++/fortran包使能LTO
- 遇到无法解决的issue再单包disable

单包初步试验结果



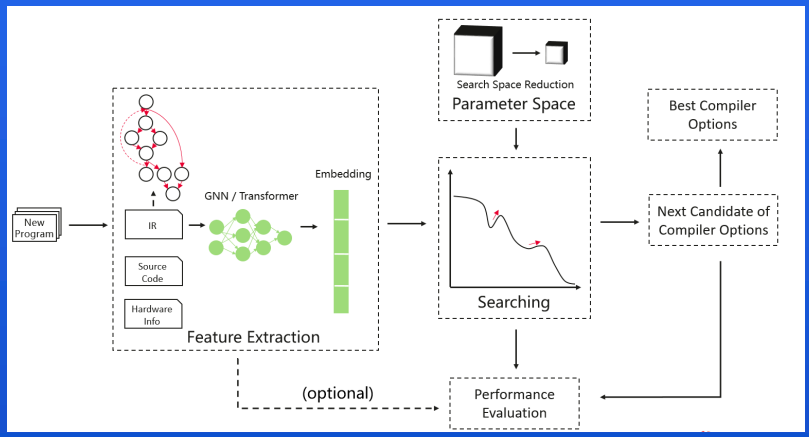
优势

- 更小的软件包体积
- 更优的开箱性能

AI4compiler——编译选项自动调优

策略

- 结合AI，实现对白盒内容的编码与代码结构特征识别
- 优化搜索算法与参数搜索空间



优势

- 白盒分析代码特征，缩短应用的调优时长
- 加速迁移对不同环境和硬件底座的调优

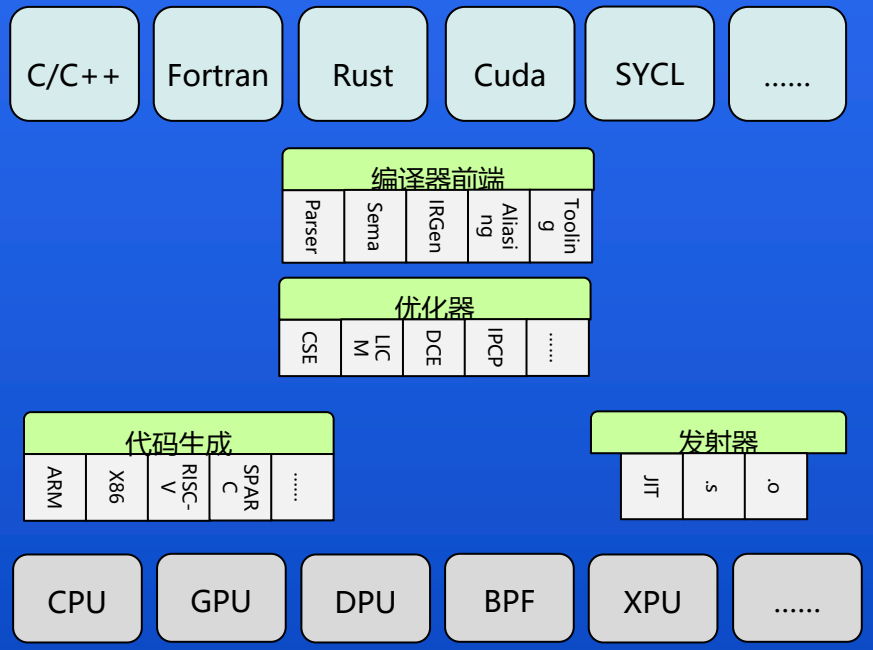
LLVM for openEuler 进展介绍及未来规划

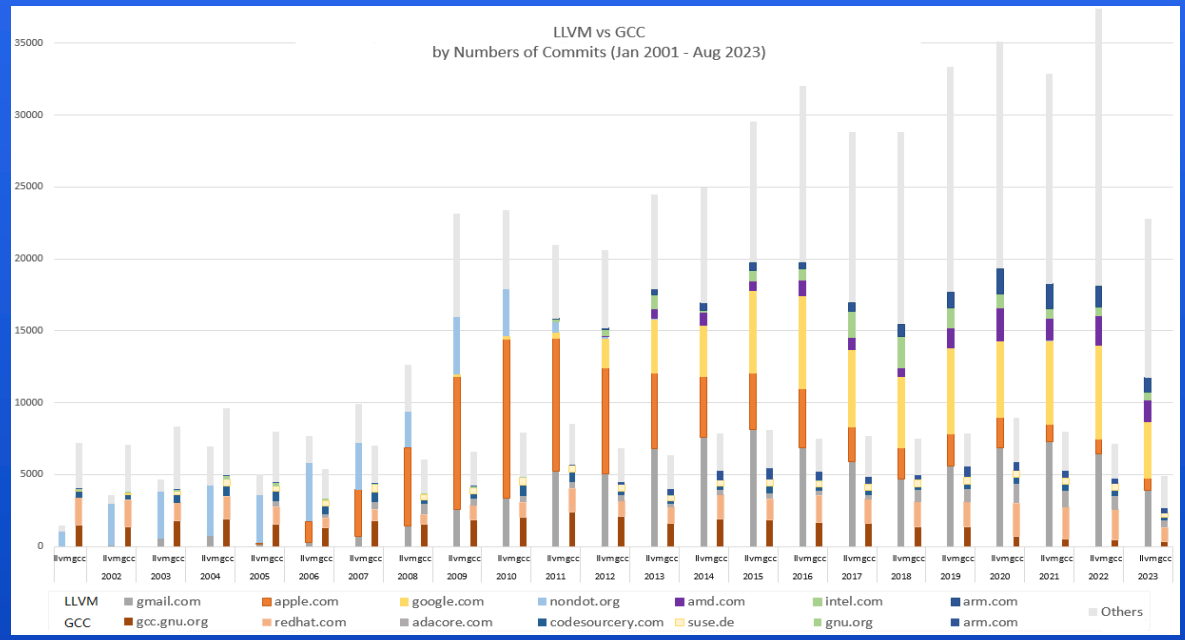
LLVM：生态及创新友好型编译基础设施，社区活跃

① **架构更先进**：模块化架构解耦，统一的IR表示，强大的Pass系统吸引着开发者投入贡献

② **协议更友好**：使用Apache License，License对各个厂商来说更加友好

③ **贡献更方便**：基于新版本C++编写，代码易读性强；完善的文档体现，更方便获得帮助





创新：JIT形态、MLIR、Clang-tidy，新语言/异构算力接入、SYCL、MLGO(AI)、BOLT、Circt... ...

社区：活跃度极高，利于技术共享与协同创新

LLVM：与openEuler协同创新，开箱即用

与操作系统形成合力，发挥基础软件协同优势

政府　安平　金融　电力　运营商　互联网

使能千行百业应用

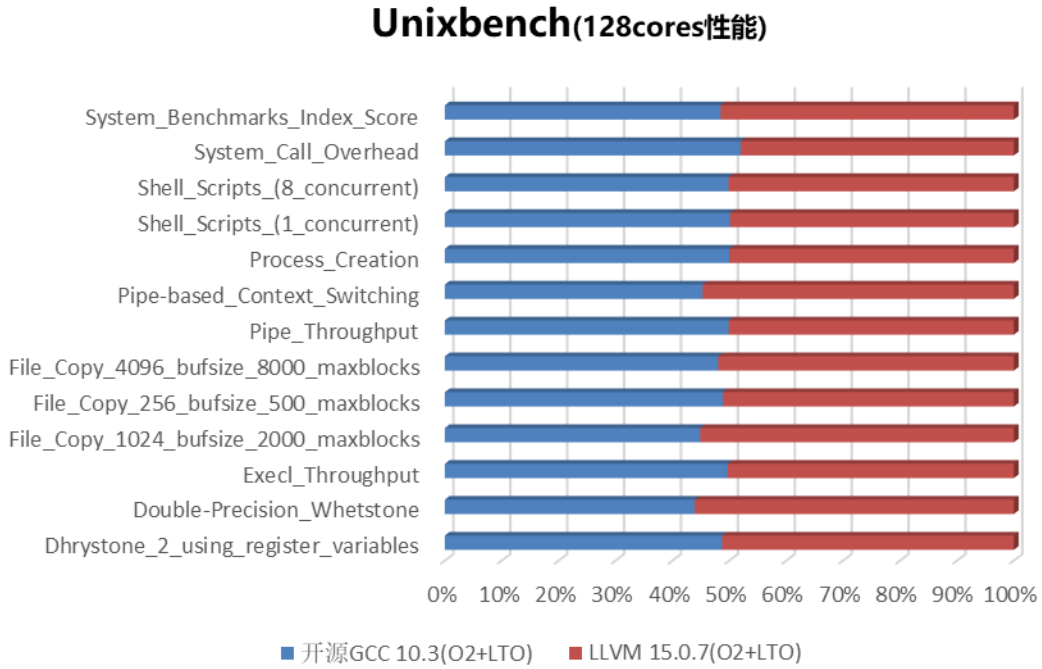
内核调优　CFGO　软硬协同

openEuler ＋ LLVM编译器

LTO优化　向量化　Autotuner

实现多架构协同运行

ARM　x86　...

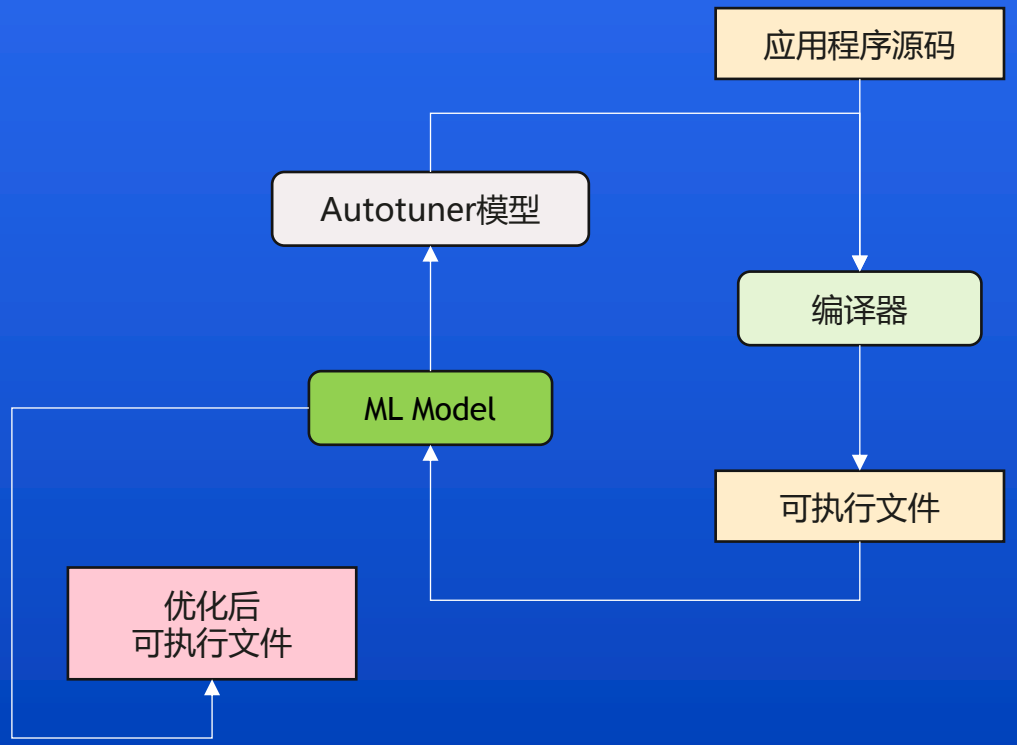


- **提升效果：Unixbench** 128cores场景下，LLVM性能相比上游开源GCC整体提升**6.35%**；（基于openEuler 22.03 LTS SP2测试）

Autotuner：基于LLVM的自动调优器，通过AI辅助调优

使用AI/Model辅助Autotuner，通过运行时反馈得到最优编译选项

Autotuning 主要用**不同的编译选项编译应用程序**，通过运行时效果反馈，作为调整优化选项依据。



AI/ML 在Autotuner中的体现

- Autotuning模型是一个基于**运行时数据的推荐系统**
- 该推荐系统接收应用程序运行时抓取的数据，然后使用**反馈的方式加速寻找最优的编译选项组合**
- 通常情况下Autotuning中的AI算法基于强化学习的算法**改进搜索策略，缩少搜索空间，大幅加速调优过程**

LLVM：高效安全开发，完善的安全选项，提升代码可靠性

➢ 高效安全开发

- 更严格的编译告警/报错
- 静态检查工具
- 运行时检查Sanitizer – 内存、线程安全等
- Safestack更强的栈保护 – 内存安全

代码复读过程中低级错误降低到原来的**10%**以下

编译过程	静态扫描工具	运行时检查
搭建LLVM构建工程： 通过LLVM编译更严格的代码检查提前发现问题	clang-tidy工具： 静态代码检查，辅助代码检视，后续支持定制检查项。	引入sanitizer工具： 关键节点执行，越界访问，内存泄漏等运行问题辅助定位。

```
void func() {
  int *p;
  p > 0;
}
```

```
error: ordered comparison between pointer and zero ('int *' and 'int')
  p > 0;
  ~ ^ ~
```

```
void test() {
  int x;
  int y = x + 1; // warn: left operand is garbage
}
```

```
void test() {
  int i, a[10];
  int x = a[i]; // warn: array subscript is undefined
}
```

```
==91527==ERROR: AddressSanitizer: stack-buffer-overflow on address 0xffffb990004c at pc 0xaaaab528ada
0 bp 0xffffe4241120 sp 0xffffe4241138
WRITE of size 4 at 0xffffb990004c thread T0
    #0 0xaaaab528ad9c  (/home/yangyang/Market_workload/wuhuasuo/loop/a.out+0x102d9c)
    #1 0xffffbc1bdadc  (/lib64/libc.so.6+0x20adc) (BuildId: 4edae11bf5b615355e734969779bde3840a6b1af)
    #2 0xaaaab51b0884  (/home/yangyang/Market_workload/wuhuasuo/loop/a.out+0x28884)

Address 0xffffb990004c is located in stack of thread T0 at offset 76 in frame
    #0 0xaaaab528ac44  (/home/yangyang/Market_workload/wuhuasuo/loop/a.out+0x102c44)

  This frame has 1 object(s):
    [32, 72) 'a' (line 2) <== Memory access at offset 76 overflows this variable
```

➢ 控制流完整性保护（前后向CFI，Kernel CFI）

```
void callee() {...}

void caller() {
  typedef void (*fp)();
  fp = callee;
  // ttacker overwrite fp
  fp();
}
```

```
callee:
  push %rbp
  ...
caller:
  ...
  mov callee, %rax
  call *%rax
  ...
```

```
.word 0x12345678
callee:
  push %rbp
  ...
caller:
  ...
  mov callee, %rax
  cmp $0x12345678,-0x4(%rax)
  bne abort
  call *%rax
  ...
```

KCFI原理：针对操作系统kernel函数的前向程序流完整性保护，用于防止间接函数调用非法跳转到函数体的内部。

选项： -fsanitize=kcfi

```
int foo() {
  return bar() + 1;
}
```

```
stp     x29, x30, [sp, #-16]!
mov     x29, sp
bl      bar
add     w0, w0, #1
ldp     x29, x30, [sp], #16
ret
```

```
str     x30, [x18], #8
stp     x29, x30, [sp, #-16]!
mov     x29, sp
bl      bar
add     w0, w0, #1
ldp     x29, x30, [sp], #16
ldr     x30, [x18, #-8]!
ret
```

SCS(后向CFI)：shadow call stack目的是为了作为 -fstack-protector 的更强替代方案。它能防止返回地址受到非线性溢出和错误的内存写操作的影响。

选项： -fsanitize=shadow-call-stack

基于LLVM构建的openEuler Embedded场景进展

目前进展：ARM64平台已支持Clang/LLVM构建版本

代码体积：

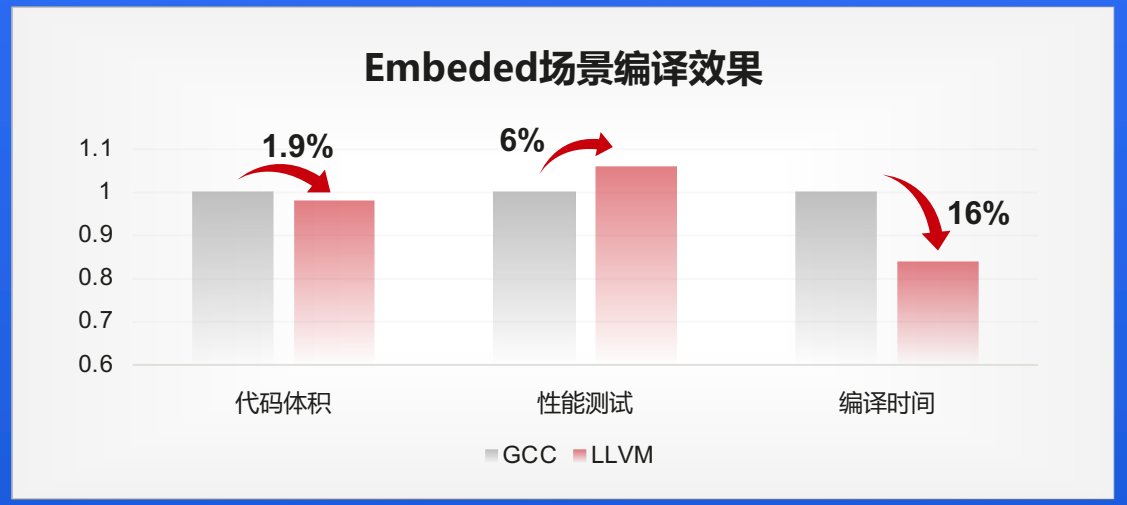
- Kernel： **llvm优于上游开源gcc 1.29%**
- 其他软件包codesize总和：**llvm优于上游开源gcc1.9%**
  - 544个可执行文件，263个文件size，llvm优于gcc，248持平
  - 282个so文件，84个文件size，llvm优于gcc，157持平

编译时间：

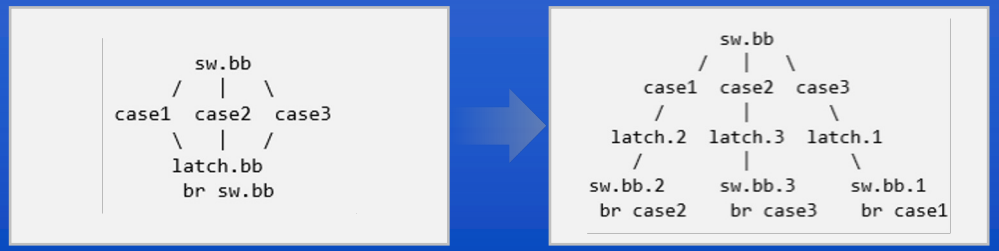
- gcc： 2483-tasks 耗时 2033.05s
- llvm：2505-tasks 耗时 1751.05s
- **约 16%的速度提升**

性能测试：

- Image用gcc构建，Coremark分别用llvm，上游gcc编译，**性能提升6%；**
- Image和Coremark分别用llvm，上游gcc编译构建，**性能提升6%；**



优化示例（Jump Threading优化）



Coremark性能提升25+%

LLVM for openEuler未来规划

版本升级及多版本计划

主版本

LLVM (12)

|-- clang

|-- llvm

多版本：

LLVM15

|-- clang-15

|-- llvm-15

|-- lld-15

主版本

LLVM-15

|-- clang-15

|-- llvm-15

|-- lld-15

|-- compiler-rt-15

|-- libcxx-15

|-- llvm-libunwind-15

|-- llvm-bolt-15

|-- mlir-15

|-- polly-15

|-- openmp-15

|-- fortran-15

|-- lldb

多版本

LLVM (12)

|-- clang

|-- llvm

主版本

LLVM（17)

|-- clang

|-- llvm

|-- lld

|-- compiler-rt

|-- libcxx

|-- llvm-libunwind

|-- llvm-bolt

|-- mlir

|-- polly

|-- openmp

|-- fortran

|-- lldb

LLVM for openEuler未来规划

LLVM平行宇宙计划（建议）

- 基于24.03 LTS版本持续构建。基于长期维护版本可以提供功能、兼容性、性能的稳定对比对象；
- 申请进入openEuler preview仓库（待向社区申请同意）。

- 首版本ISO发布

- 更多的包基于LLVM构建
- LTO+PGO泛化能力（kernel）

- 更多的包基于LLVM构建
- LTO+PGO泛化能力（其他重点包）

资源获取与社区支持

交付件类型	链接	使用说明
软件包	https://www.hikunpeng.com/developer/devkit/compiler	毕昇编译器页签点击”毕昇编译器软件包下载”，下载后解压使用
	https://oepkgs.net/	搜索“bisheng-compiler”关键字
文档	https://www.hikunpeng.com/document/detail/zh/kunpengdevps/compiler/ug-bisheng/kunpengbisheng_06_0001.html	毕昇编译器用户指南
问题讨论	https://www.hikunpeng.com/forum/forum-0105101360563095011-1.html	鲲鹏社区论坛，可发帖提问

扫码关注毕昇编译公众号

加小助手进Compiler交流群

THANKS

THANKS

THANKS