PR #45243 完整报告

vllm-project/vllm

[RISC-V] Enable BF16 on VLEN=256 hardware

合并时间：2026-07-06 14:05

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/45243

执行摘要

- 一句话：RISC-V BF16 强制启用与 VLEN=256 修复
- 推荐动作：建议精读 BF16Vec32 构造函数的改动，理解 RISC-V LMUL 扩展与元素复制的技巧。测试方面应增加 VLEN=256 硬件上的 BF16 算子单元测试，以及确保 VLEN=128 回归测试通过。

功能与动机

早期的 PR #36578 通过 find_isa 检测 /proc/cpuinfo 中的 zvfbfmin 标志来决定是否启用 BF16。但最新的 RISC-V 硬件（如 Spacemit X100 / K3）由于内核 / 固件限制，不会在 /proc/cpuinfo 中报告 zvfbfmin，导致 BF16 被静默禁用。本 PR 提供强制启用的逃生舱机制，并修复因 LMUL 预留不足导致的编译或运行时问题。

实现拆解

1. 新增环境变量强制启用 BF16— 在 cmake/cpu_extension.cmake 中读取 VLLM_CPU_RVV_BF16 环境变量，若设置则无条件设置 RVV_BF16_FOUND=ON，绕过 /proc/cpuinfo 检测。该变量为 unchecked override，用户需确保硬件真正支持 zvfbfmin，否则会触发 SIGILL。
2. 修复 BF16Vec32 构造函数中 LMUL 分配— 在 csrc/cpu/cpu_types_riscv_impl.hpp 中，原有实现直接使用 __riscv_vcreate_v_u16 从 LMUL_128 扩展为 LMUL_512，这在 VLEN=256（LMUL 比例变化）下会因类型不匹配导致编译失败。新实现通过两步展开：先用 vlmul_ext 将 LMUL_128 提升为 LMUL_256，然后通过 vslideup 复制元素构建 16 元素 half，最后用 vcreate 双倍到 LMUL_512。
3. 改进 cmake 错误提示— 当无法自动检测 VLEN 时，提示用户通过 CMAKE_ARGS 指定 VLLM_RVV_VLEN，示例更清晰。

关键文件：

- csrc/cpu/cpu_types_riscv_impl.hpp（模块 CPU 内核；类别 source；类型 core-logic；符号 BF16Vec32::BF16Vec32(const BF16Vec8&)）：核心源码文件，修复 BF16Vec32 构造函数中 LMUL 分配错误，新增 vslideup 元素复制逻辑。
- cmake/cpu_extension.cmake（模块 构建脚本；类别 other；类型 configuration）：构建系统配置，新增 VLLM_CPU_RVV_BF16 环境变量支持，优化 VLEN 检测错误提示。

关键符号：BF16Vec32::BF16Vec32(const BF16Vec8&)

关键源码片段

csrc/cpu/cpu_types_riscv_impl.hpp

核心源码文件，修复 BF16Vec32 构造函数中 LMUL 分配错误，新增 vslideup 元素复制逻辑。

```
// 修复前：直接 vcreate 从 LMUL_128 跳到 LMUL_512，在 VLEN=256 下类型不匹配
// 修复后：分两步扩展 —— 先提升到 LMUL_256，再用 vslideup 复制元素，最后倍增至 LMUL_512

explicit BF16Vec32(const BF16Vec8& v) {
  fixed_u16x8_t u16_val = bf16_to_u16(v.reg);

  // 第一步：将 LMUL_128 提升至 LMUL_256（类型一致后 vslideup 操作合法）
  // VLEN=256 时：mf2 → m1；VLEN=128 时：m1 → m2
  fixed_u16x16_t ext =
      RVVI4(__riscv_vlmul_ext_v_u16, LMUL_128, _u16, LMUL_256)(u16_val);

  // 第二步：在 LMUL_256 向量中放置左右各 8 个元素
  fixed_u16x16_t half = RVVI(__riscv_vmv_v_x_u16, LMUL_256)(0, 16); // 全零
  half = RVVI(__riscv_vslideup_vx_u16, LMUL_256)(half, ext, 0, 8); // 低 8 位
  half = RVVI(__riscv_vslideup_vx_u16, LMUL_256)(half, ext, 8, 16); // 高 8 位

  // 第三步：LMUL_256 → LMUL_512 翻倍（VLEN=256: m1→m2, VLEN=128: m2→m4）
  fixed_u16x32_t dst =
      RVVI4(__riscv_vcreate_v_u16, LMUL_256, _u16, LMUL_512)(half, half);

  reg = RVVI4(__riscv_vreinterpret_v_u16, LMUL_512, _bf16, LMUL_512)(dst);
};
```

cmake/cpu_extension.cmake

构建系统配置，新增 VLLM_CPU_RVV_BF16 环境变量支持，优化 VLEN 检测错误提示。

```
# 读取用户设置的环境变量
set(ENABLE_RVV_BF16 $ENV{VLLM_CPU_RVV_BF16})

# ... 在 RVV_FP16_FOUND 检测块结束后，插入强制覆盖逻辑
# 某些内核（如 Bianbu on Spacemit X100）不在 /proc/cpuinfo 中报告 zvfbfmin
# 但硬件实际支持，通过环境变量 VLLM_CPU_RVV_BF16=1 强制启用
if (ENABLE_RVV_BF16)
  set(RVV_BF16_FOUND ON)
  message(STATUS "RVV BF16 support enabled via VLLM_CPU_RVV_BF16 environment variable")
endif()
```

另外优化了 VLEN 未检测时的错误信息，明确提示通过 CMAKE_ARGS 设置 VLLM_RVV_VLEN。

评论区精华

无 review 评论。PR 作者 velonica0 请求维护者 bigPYJ1151 审查，bigPYJ1151 直接批准。

- 暂无高价值评论线程

风险与影响

- 风险：
  1. 运行时风险：VLLM_CPU_RVV_BF16=1 是 unchecked override，如果用户在不支持 zvfbfmin 的硬件上启用，会因非法指令（SIGILL）崩溃。文档中已说明此风险，但未提供运行时校验。
  2. VLEN=128 兼容性：改动显式考虑了 VLEN=128 路径（注释说明 vslideup 在 VLEN=128 下为 m1→m2），但未添加测试验证。
  3. 回归风险：BF16Vec32 构造逻辑完全重写，可能影响所有 VLEN=128/256 的 RISC-V 平台。 - 影响：影响范围：仅 RISC-V CPU 后端，仅影响 BF16 数据类型的使用者。 影响程度：中等。对 Spacemit X100 等硬件用户是功能修复（BF16 从不可用到可用）；对其他 RISC-V 用户无影响（默认仍通过 /proc/cpuinfo 检测）。 用户可见性：需设置环境变量才能启用，非默认行为。
- 风险标记：无测试覆盖，手动强制启用风险，平台特定

关联脉络

- PR #36578 Enable BF16 via /proc/cpuinfo detection: 本 PR 解决的问题来源：该 PR 添加的 /proc/cpuinfo 检测在新硬件上失效。