PR #44478 完整报告

vllm-project/vllm

[CPU][RISC-V] Enable oneDNN W8A8 INT8 to run on RISC-V

合并时间：2026-06-11 12:09

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/44478

执行摘要

此 PR 在 vLLM 的 CPU 后端中为 RISC-V 架构启用了 oneDNN 的 W8A8 INT8 量化路径。通过四步最小化改动（新增向量类型、扩展条件编译、调整 CMake 构建），解决了 RISC-V 上加载 W8A8 量化模型即崩溃的问题。改动量小，但为 RISC-V 用户解锁了一个关键的量化推理模式。

功能与动机

在 RISC-V 上加载压缩感知 W8A8 量化模型会立即崩溃。根本原因在于 CPU W8A8 调度器（vllm/model_executor/kernels/linear/scaled_mm/cpu.py）的oneDNN分支仅针对x86（SGL）、AArch64 和 POWER 编译，RISC-V 被排除在外。该 PR 打通了从 CMake 编译门到 C++ 算子注册再到 RVV 向量类型的全链路。

实现拆解

1. 新增 int8 向量类型：在 csrc/cpu/cpu_types_riscv_defs.hpp 中添加 fixed_i8x16_t（16 个 int8 元素的固定向量），对应 RVV LMUL=128。
2. 实现 INT8Vec16 类：在 csrc/cpu/cpu_types_riscv_impl.hpp 中新增 INT8Vec16，提供从 FP32Vec16 到 INT8 的量化转换（VFCVT + 两级 VNCLIP）和向量化存储。同时为 FP32Vec16 增加带元素数参数的 max/min 重载，支持可变向量长度操作。
3. 扩展算子注册条件：在 csrc/cpu/torch_bindings.cpp 中将 oneDNN 相关算子的条件编译宏扩展为包含 defined(__riscv_v)，使 release_dnnl_matmul_handler、onednn_mm、onednn_scaled_mm 等算子在 RISC-V 上注册。
4. 调整 CMake 构建：在 cmake/cpu_extension.cmake 中，当 VLLM_RVV_VLEN 被定义但非法时主动报错；将 oneDNN 的构建条件从仅 x86/ARM/POWER 扩展到检测到 RVV_FP16 或 RVV_BF16 时也启用。

csrc/cpu/cpu_types_riscv_impl.hpp

新增 INT8Vec16 类和 FP32Vec16 的 elem_num 参数重载，是量化路径的核心向量类型。

```
// INT8Vec16: 将 FP32 向量量化为 INT8 并存储
struct INT8Vec16 : public Vec<INT8Vec16> {
  constexpr static int VEC_ELEM_NUM = 16;
  fixed_i8x16_t reg;

  // 从 FP32Vec16 转换：VFCVT→VNCLIP(w, 0, RN)→VNCLIP(w, 0, RN)
```

```
  explicit INT8Vec16(const FP32Vec16& vec) {
    auto i32_vec =
        RVVI(__riscv_vfcvt_x_f_v_i32, LMUL_512)(vec.reg, VEC_ELEM_NUM);
    auto i16_vec = RVVI(__riscv_vnclip_wx_i16, LMUL_256)(
        i32_vec, 0, __RISCV_VXRM_RNU, VEC_ELEM_NUM);
    reg = RVVI(__riscv_vnclip_wx_i8, LMUL_128)(i16_vec, 0, __RISCV_VXRM_RNU,
                                               VEC_ELEM_NUM);
  }

  // 存储全部 16 个元素
  void save(int8_t* ptr) const {
    RVVI(__riscv_vse8_v_i8, LMUL_128)(ptr, reg, VEC_ELEM_NUM);
  }
  // 存储前 elem_num 个元素
  void save(int8_t* ptr, int elem_num) const {
    RVVI(__riscv_vse8_v_i8, LMUL_128)(ptr, reg, elem_num);
  }
};

// FP32Vec16 增加可变元素数重载，与固定 VEC_ELEM_NUM 互备
FP32Vec16 max(const FP32Vec16& b, const int elem_num) const {
  return FP32Vec16(
      RVVI(__riscv_vfmax_vv_f32, LMUL_512)(reg, b.reg, elem_num));
}
FP32Vec16 min(const FP32Vec16& b, const int elem_num) const {
  return FP32Vec16(
      RVVI(__riscv_vfmin_vv_f32, LMUL_512)(reg, b.reg, elem_num));
}
```

csrc/cpu/torch_bindings.cpp

修改条件编译宏，使 oneDNN 算子在 RISC-V 上注册。

```
// 之前：仅 x86、AArch64 或 POWER 编译 oneDNN 算子
// 之后：增加 RISC-V RVV 条件
#if defined(__AVX512F__) || defined(__AVX2__) || \
    (defined(__aarch64__) && !defined(__APPLE__)) || defined(__powerpc64__) || \
    defined(__riscv_v) /* <-- 新增 RISC-V 分支 */
  // 以下 oneDNN 相关算子在满足条件时注册
  ops.def("release_dnnl_matmul_handler(int handler) -> ()", &release_dnnl_matmul_handler);
  ops.def("create_onednn_mm_handler(...) -> int", &create_onednn_mm_handler);
  ops.impl("onednn_mm", torch::kCPU, &onednn_mm);
  ops.def("is_onednn_acl_supported() -> bool", &is_onednn_acl_supported);
  ops.def("create_onednn_scaled_mm_handler(...) -> int", &create_onednn_scaled_mm_handler);
  ops.def("onednn_scaled_mm(...) -> ()");
  ops.impl("onednn_scaled_mm", torch::kCPU, &onednn_scaled_mm);
#endif
```

评论区精华

无 review 评论，直接由维护者批准合并。

风险与影响

- 性能风险：初步测试显示生成速度仅 0.245 token/s，远低于 x86 水平，但功能正确。
- 编译配置风险：VLLM_RVV_VLEN 的编译时设定需与硬件匹配；CMake 中要求 RVV_FP16 和 RVV_BF16 同时检测才启用 oneDNN，可能误判部分硬件。
- 精度风险：INT8Vec16 使用 vnclip 进行窄化，遵循标准的 __RISCV_VXRM_RNU 舍入模式，符合预期。
- 影响范围：仅 RISC-V 平台受影响，其他架构无变化。

关联脉络

无直接关联的历史 PR。该 PR 是 vLLM CPU 后端支持 RISC-V 系列工作的一部分，未来可能需要优化性能、增加 CI 覆盖。