PR #46739 完整报告

vllm-project/vllm

[CPU][BugFix] Multiple fixes to w4a8_int8 CPU MoE path

合并时间：2026-07-06 13:39

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/46739

执行摘要

- 一句话：修复 CPU INT4 MoE 的多项 bug 并恢复通道量化支持
- 推荐动作：本 PR 属于中等重要度的 Bugfix + 重构，对 Arm CPU 用户至关重要。建议阅读以下设计点：
  1. 将硬件特有检查迁移到 is_supported_config：这种模式保持了构造函数与平台无关，允许后端在配置阶段优雅降级。
  2. Fake 实现的注册方法：对于 torch.compile 支持，查看 vllm/_custom_ops.py 中的 register_fake 用法。
  3. 测试重构：convert_to_w4a8_int8_moe_format 统一了权重打包逻辑，测试职责更清晰。

整体变更质量较高，审核中关键问题都已解决并获得 Approve。

功能与动机

Purpose w4a8 MoE refactor in #42789 broke parts of this path. This PR fixes the group size passed to C++ kernel, registers fake impl for torch compile, adds support for fp32, bf16, fp16 pre quant activations, and restores channelwise support which was dropped by previous refactor.

实现拆解

1. 修复 C++ 内核函数签名与分组大小传递：在 csrc/moe/dynamic_4bit_int_moe_cpu.cpp 中移除冗余的 I2 参数，将参数名 H/I 改为 hidden_size/intermediate_size，避免 lint 警告；修正 group_size 传递给 _dyn_quant_matmul_4bit 的逻辑，确保 channel-wise 时使用正确的有效分组大小。
2. 注册 torch.compile fake 实现：在 vllm/_custom_ops.py 中为 dynamic_4bit_int_moe 算子添加 @register_fake 修饰的 dynamic_4bit_int_moe_fake 函数，使 torch.compile 能够在不执行实际内核的情况下完成图构建。
3. 恢复通道量化支持：在 vllm/model_executor/layers/quantization/utils/quant_utils.py 中新增 kInt4W4A8StaticChannelSym 量化键，并在 cpu_int4_moe.py 的 _supports_quant_scheme 中将其加入支持的量化方案列表。
4. 扩展 pre-quant activation dtype 支持：在 C++ 内核中增加根据输入 dtype 自动转换为 fp32 的逻辑，使 kernel 能处理 fp32、bf16、fp16 三种预量化激活输入。同时将 gates 张量的 dtype 与输入保持一致，避免不必要的 cast，并将最终输出转换为原始 dtype。

5. 重构 is_supported_config 与 CompressedTensorsW4A8Int8MoEMethod.__init__：将 Arm 内核特有的检查（架构检测、dtype 验证、PyTorch op 可用性）从 __init__ 迁移到 CPUExpertsInt4.is_supported_config 静态方法中，使构造函数保持平台无关。在 compressed_tensors_moe_w4a8_int8.py 中移除冗余的 QuantizationStrategy 校验和 _dyn_quant_* op 检查（已由 is_supported_config 覆盖），并增加分组大小整除性验证。
6. 更新测试并加入 Arm CI：在 tests/kernels/moe/test_cpu_int4_moe.py 中移除内联的 _pack_int4_weight_to_kleidi 函数，改用 convert_to_w4a8_int8_moe_format 统一打包；添加 ref_int4_moe 参考实现以加速验证；将模块级跳过条件收紧为仅 Arm CPU；在 .buildkite/scripts/hardware_ci/run-cpu-test-arm.sh 中加入该测试的执行命令。

关键文件：

- vllm/model_executor/layers/fused_moe/experts/cpu_int4_moe.py（模块 MoE 层；类别 source；类型 core-logic；符号 is_supported_config, _activation_kind）：核心改动：新增 is_supported_config 静态方法，将硬件特有检查从构造函数分离；恢复通道量化支持，扩展激活类型支持；修改 _activation_kind 为静态方法。
- vllm/model_executor/layers/quantization/compressed_tensors/compressed_tensors_moe/compressed_tensors_moe_w4a8_int8.py（模块 量化层；类别 source；类型 refactor；符号 CompressedTensorsW4A8Int8MoEMethod.init）：移除了构造函数中冗余的 Arm 特有检查（已由 is_supported_config 覆盖），增加分组大小整除性验证，简化了初始化逻辑。
- tests/kernels/moe/test_cpu_int4_moe.py（模块 测试；类别 test；类型 test-coverage；符号 _make_int4_moe_weights, ref_int4_moe, test_cpu_int4_moe_kernel）：全面重构：使用 convert_to_w4a8_int8_moe_format 替代内联打包，新增 ref_int4_moe 参考实现，收紧跳过条件为仅 Arm CPU，提升测试可维护性和覆盖。
- csrc/moe/dynamic_4bit_int_moe_cpu.cpp（模块 C++ 内核；类别 source；类型 core-logic；符号 dynamic_4bit_int_moe_cpu）：核心 C++ 内核：移除冗余 I2 参数，重命名 H/I 为 hidden_size/intermediate_size，增加输入 dtype 自动提升逻辑，将 gates dtype 与输入对齐，修复 group_size 传递。
- vllm/_custom_ops.py（模块 算子注册；类别 source；类型 core-logic；符号 dynamic_4bit_int_moe_fake）：为 dynamic_4bit_int_moe 注册 torch.compile fake 实现，使其在编译模式下不会崩溃。
- vllm/model_executor/layers/quantization/utils/quant_utils.py（模块 量化键；类别 source；类型 data-contract）：新增 kInt4W4A8StaticChannelSym 量化键，是恢复通道量化的基础设施。
- vllm/model_executor/layers/fused_moe/oracle/w4a8_int8.py（模块 MoE 层；类别 source；类型 refactor；符号 convert_to_w4a8_int8_moe_format）：重命名变量 I2 为 w13_out_features 以绕过 lint，H 改为 hidden_size 等，保持代码风格一致。
- csrc/ops.h（模块 头文件；类别 source；类型 core-logic）：更新动态 4bit MoE 算子的 C++ 签名声明，移除 I2 参数，重命名 H/I。
- csrc/cpu/torch_bindings.cpp（模块 绑定；类别 source；类型 core-logic）：更新 torch 绑定中的参数传递，匹配新签名。

- .buildkite/scripts/hardware_ci/run-cpu-test-arm.sh（模块 CI 脚本；类别 infra；类型 infrastructure）：将 INT4 MoE 测试加入 Arm CI 执行脚本，确保回归检测。

关键符号：dynamic_4bit_int_moe_cpu, dynamic_4bit_int_moe_fake, is_supported_config, _activation_kind, _supports_quant_scheme, _make_int4_moe_weights, ref_int4_moe, convert_to_w4a8_int8_moe_format, CompressedTensorsW4A8Int8MoEMethod.init

关键源码片段

vllm/model_executor/layers/fused_moe/experts/cpu_int4_moe.py

核心改动：新增 is_supported_config 静态方法，将硬件特有检查从构造函数分离；恢复通道量化支持，扩展激活类型支持；修改 _activation_kind 为静态方法。

```
# SPDX-License-Identifier: Apache-2.0
# file: vllm/model_executor/layers/fused_moe/experts/cpu_int4_moe.py

class CPUExpertsInt4(mk.FusedMoEExpertsMonolithic):
    # ... other methods ...

    @staticmethod
    def is_supported_config(
        cls: type[mk.FusedMoEExperts],
        moe_config: FusedMoEConfig,
        weight_key: QuantKey | None,
        activation_key: QuantKey | None,
        activation_format: mk.FusedMoEActivationFormat,
    ) -> tuple[bool, str | None]:
        """Check if the current hardware and config support this ARM kernel."""
        # Only Arm CPU is supported
        if (
            not current_platform.is_cpu()
            or current_platform.get_cpu_architecture() != CpuArchEnum.ARM
        ):
            return False, "kernel only supports Arm CPU"

        # Input dtype must be one of fp32/bf16/fp16 (fp16 will be upcasted)
        if moe_config.in_dtype not in (
            torch.float32,
            torch.bfloat16,
            torch.float16,
        ):
            return (
                False,
                f"kernel does not support {moe_config.in_dtype} input/output dtype",
            )

        # Check that PyTorch 2.7+ KleidiAI ops are available
        try:
            _ = torch.ops.aten._dyn_quant_matmul_4bit
```

```
            _ = torch.ops.aten._dyn_quant_pack_4bit_weight
        except AttributeError:
            return (
                False,
                f"PyTorch {torch.__version__} does not support "
                "_dyn_quant_* 4bit ops. Install a newer version",
            )

        # Fall through to parent class generic checks
        return mk.FusedMoEExperts.is_supported_config(
            cls, moe_config, weight_key, activation_key, activation_format,
        )

    @staticmethod
    def _supports_quant_scheme(
        weight_key: QuantKey | None,
        activation_key: QuantKey | None,
    ) -> bool:
        """Supports INT4 weights with INT8 dynamic activations."""
        # channelwise or groupwise with group size being a multiple of 32
        SUPPORTED_W_A = [
            (kInt4W4A8StaticChannelSym, None), # <-- restored channel-wise support
            (kInt4W4A8StaticGroup128Sym, None),
            (kInt4W4A8StaticGroup64Sym, None),
            (kInt4W4A8StaticGroup32Sym, None),
            (kInt4W4A8StaticGroupSym, None),
        ]
        return (weight_key, activation_key) in SUPPORTED_W_A
```

csrc/moe/dynamic_4bit_int_moe_cpu.cpp

核心 C++ 内核：移除冗余 I2 参数，重命名 H/I 为 hidden_size/intermediate_size，增加输入 dtype 自动提升逻辑，将 gates dtype 与输入对齐，修复 group_size 传递。

```
// file: csrc/moe/dynamic_4bit_int_moe_cpu.cpp

torch::Tensor dynamic_4bit_int_moe_cpu(
    torch::Tensor x, torch::Tensor topk_ids, torch::Tensor topk_weights,
    torch::Tensor w13_packed, torch::Tensor w2_packed, int64_t hidden_size,
    int64_t intermediate_size, int64_t group_size,
    bool apply_router_weight_on_input, int64_t activation_kind) {

  // ...

  // _dyn_quant_matmul_4bit kernel natively supports these pre-quant activation
  // dtypes:
  // - fp32: with channelwise and groupwise
  // - bf16: with channelwise -> upcast to fp32 for groupwise
  // - fp16: not supported -> upcast to fp32 for groupwise & channelwise
  const auto output_dtype = x_c.scalar_type();
```

```
    const bool should_cast_input =
        ((group_size != -1) && output_dtype == at::kBFloat16) ||
        output_dtype == at::kHalf;
    if (should_cast_input) {
      x_c = x_c.to(at::kFloat);
    }

    // Make gates the same dtype as input to avoid extra cast
    auto gates_c = topk_weights.to(x_c.scalar_type()).contiguous();

    // ...
  }
```

评论区精华

主要讨论了以下问题：

- compress_tensors_moe_w4a8_int8.py 中 Arm 特有检查的冗余性：nikhil-arm 指出 is_supported_config 已经处理了校准，不应在构造函数中重复判断。fadara01 解释移除这些检查是为了保持构造函数平台无关，只在校后端选择时失败。最终达成一致，移除了冗余检查。
- 测试文件中对 PyTorch 版本 op 可用性检查的保留：nikhil-arm 担心移除 hasattr(torch.ops.aten, '_dyn_quant_pack_4bit_weight') 会导致旧版 PyTorch 上测试无声跳过。fadara01 认为 tests 本就只在 Arm CI 运行，移除检查能让测试在 op 缺失时失败，以便发现回归。nikhil-arm 最终认同。
- C++ 内核中避免不必要的 to() 转换：nikhil-arm 建议通过提前分配正确 dtype 的 expert_gates 来消除多余的 cast。fadara01 采纳并调整了 gates_c 的创建。
- 变量命名与 lint：bigPYJ1151 注意到 I2 被重命名为 w13_out_features，fadara01 证实这是为了绕过 lint 检查。
- 移除构造函数中 Arm 特有检查 (design)：已移除冗余检查，构造函数保持平台无关。
- 测试中 PyTorch op 可用性检查 (testing)：移除 op 可用性检查，测试仅在 Arm CI 运行，op 缺失时测试会因调用失败而显式报错。
- C++ 内核中消除多余 cast (performance)：已通过将 gates_c 的 dtype 设置为与 x_c 一致来解决，消除了多余的 cast。
- 变量重命名以绕过 lint (style)：已完成重命名，I2 改为 w13_out_features，H 改为 hidden_size 等。

风险与影响

- 风险：主要技术风险包括：
  - C++ 内核数据类型转换：新增的 should_cast_input 逻辑将 bf16/fp16 输入强制转换为 fp32 再传递给 _dyn_quant_matmul_4bit。虽然现在 kernel 原生支持 fp32 和 bf16（channel-wise），但 fp16 会被全部 upcast，可能带来轻微的精度损失和性能开销。测试中未覆盖所有 dtype 组合。

- 通道量化恢复的数值验证：通道量化已被之前重构丢弃，本次恢复后未在 E2E 测试中充分验证数值正确性，仅单元测试使用随机权重。
- PyTorch 版本兼容性：新代码依赖 torch.ops.aten._dyn_quant_matmul_4bit 和 _dyn_quant_pack_4bit_weight，这些算子仅在 PyTorch 2.7+ 的 Arm 构建中可用。is_supported_config 中虽有 try-except 捕获，但若在旧版本上加载已量化模型，错误信息可能不够直观。
- 测试仅在 Arm CI 运行：虽然添加了 Arm CI 执行，但非 Arm 平台完全跳过，可能遗漏因平台相关代码变更导致的构建或链接错误。
- 影响：用户影响：对使用 vLLM 在 Arm CPU 上进行 w4a8 量化 MoE 推理的用户有直接正面影响——恢复了通道量化支持，扩展了激活数据类型支持（fp32/bf16/fp16），并修复了分组大小传递错误，从而提升了量化模型的可用性和推理质量。

系统影响：C++ 内核签名变更（移除 I2）需所有调用点同步更新，但实际只有 torch_bindings.cpp 一处调用，风险可控。Fake 实现的注册保证了 torch.compile 模式下图构建不崩溃。

团队影响：代码结构更清晰——Arm 特有检查从构造函数集中到 is_supported_config 接口，便于后续其他后端复用 CompressedTensorsW4A8Int8MoEMethod。测试改用统一打包函数，减少维护成本。

- 风险标记：内核路径变更，激活类型转换，依赖 PyTorch 新算子，测试仅 Arm CI

关联脉络

- PR #42789 [Refactor] w4a8 MoE refactor: 本 PR 旨在修复 #42789 重构引入的回归，例如组大小传递错误、通道量化丢失等问题。