

PR #54042 完整报告

vllm-project/vllm

[Bugfix][CPU] Fix several bugs

合并时间: 2026-08-31 15:33

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/54042>

执行摘要

- 一句话: CPU 后端多 bug 修复: MoE 溢出、小 GEMM、SWA 块大小
- 推荐动作: 建议精读。虽然 PR 标题平平无奇, 但内部包含三类值得学习的修复模式: 一是「int32 截断 + size_t 符号扩展」导致分配失败的调试思路, 二是 C++ 算子「返回 Tensor → 预分配 out」的契约演变如何降低内存管理开销, 三是「builder 统一计算并下 ISA 到层」取代「每层读全局配置」的一致性设计。风险集中在 weight_packed_linear 的 ABI 变更与 register_fake 移除, 后续启用 CPU torch.compile 前需要补回 meta 实现。

功能与动机

PR body 列出四个明确目标: 修复 CPU fused MoE buffer size 溢出、修复 Qwen3.6 router logits GEMM 问题、修复纯 SWA 模型的 block_size 问题、抑制噪音日志。其中溢出问题的根因在提交 c27e8c2 中有详细描述: fused_moe_impl 的 scratch buffer 尺寸计算全程使用 int32_t, 而 Qwen3-235B-A22B 在 TP4、max_num_batched_tokens=16384 时单个缓冲区字节数可达 2^{31} ; 赋值截断为负值后符号扩展到 size_t 约 2^{64} , 导致 aligned_alloc 失败。GEMM 问题则源于小权重 (MoE router 等) 路径的 packed weight 可能是 float32, 原分派逻辑与 C++ 实现契约不一致。

实现拆解

第一步: C++ 内核层 —— 修复 fused MoE scratch buffer 溢出

csrc/cpu/cpu_fused_moe.cpp 的 fused_moe_impl 中, 所有缓冲区大小与偏移量 (common_buffer_offset、w13_gemm_output_buffer_size、w2_gemm_output_buffer_size、buffer_size 等) 从 int32_t 改为 int64_t, 乘法处显式 static_cast 提升。布局语义不变, 仅放大中间类型的表示范围, 避免大 batch 下字节数越过 INT32_MAX 后截断为负值、再符号扩展成约 2^{64} 字节传给 ScratchPadManager::realloc(size_t) 而触发 aligned_alloc 失败。

第二步: C++ GEMM 契约调整 —— weight_packed_linear 改为预分配输出

csrc/cpu/sgl-kernels/gemm.cpp 将 weight_packed_linear 从「返回 Tensor」改为「void + 预分配 out 参数」, 并 TORCH_CHECK(is_vnni) 强制走预打包权重路径, 不再内部调用 convert_weight_packed。csrc/cpu/torch_bindings.cpp 同步更新 TORCH_LIBRARY schema 为 (out, mat1, mat2, bias, is_vnni) -> ()。原因: 原实现只能在 2D 输入上返回 2D 输出, 且内部打包逻辑与调用方职责不清; 新契约把打包职责上移到 Python 层, 并支持任意前导维度的输入。

第三步：Python 封装与 GEMM 分派修复

vllm/_custom_ops.py 移除与旧签名绑定的 register_fake("_C::weight_packed_linear"), 新增 weight_packed_linear_cpu(x, packed_weight, n, bias): 内部预分配 output、展平输入后调用底层 op, 向调用方提供简洁接口。vllm/model_executor/layers/utils.py 的 dispatch_cpu_unquantized_gemm 小 GEMM 分支改为「先 convert_weight_packed 预打包、把 layer.bias 就地转 float32、闭包捕获 ops.weight_packed_linear_cpu」, 修复 Qwen3.6 router logits GEMM 的分派错误。

第四步：CPU 注意力后端 —— block_size 与 ISA 传播

vllm/v1/attention/backends/cpu_attn.py 三处联动修改:

get_supported_kernel_block_sizes 从 MultipleOf(16) 改为 MultipleOf(32), 对齐 SGL CPU 内核的硬件对齐要求; CPUAttentionMetadataBuilder.__init__ 中 self.block_size 从全局 vllm_config.cache_config.block_size 改为 kv_cache_spec.block_size, 修复纯 SWA 模型注意力组的块大小校验失败; 新增 _set_isa_to_layers 在 metadata 构建阶段统一把 ISA 写回各 layer, CPUAttentionBackendImpl.__init__ 中基于 get_current_vllm_config() 的 ISA 计算被删除, forward 与 do_kv_cache_update 改读 layer.isa, 消除各层自行推算 ISA 的时序不一致。

第五步：平台适配与测试配套

vllm/model_executor/layers/mamba/gdn/qwen_gdn_linear_attn.py 在 CPU 平台提前返回并记录 CPU 内核日志; gdn_decode_kernel == "cuda" 增加 current_platform.is_cuda_alike() 守卫, CPU 上直接置为 "CPU", 避免误触发 CUDA 专属检查。vllm/triton_utils/importing.py 把 Triton CPU backend 检查提前到活跃驱动校验之前, 并将 CPU driver 补入 active_drivers, 抑制「0 个活跃驱动」误报导致 Triton 被禁用。测试侧: tests/conftest.py 新增 _default_block_size (CPU=128、XPU=64、其他=16); KV connector 单元测试的 block_size 从 16 改为 32; tests/v1/attention/test_group_head_counts.py 小范围适配。

关键文件:

- vllm/_custom_ops.py (模块 自定义算子; 类别 source; 类型 core-logic; 符号 weight_packed_linear_cpu, weight_packed_linear_fake): 新增 weight_packed_linear_cpu 包装函数, 将 C++ 端 weight_packed_linear 的「预分配输出」契约封装为与调用方匹配的 Python 接口, 同时移除不再匹配新签名的 register_fake 实现。
- vllm/v1/attention/backends/cpu_attn.py (模块 注意力后端; 类别 source; 类型 core-logic; 符号 _set_isa_to_layers, CPUAttentionMetadataBuilder, CPUAttentionBackendImpl, get_supported_kernel_block_sizes): 修复纯 SWA 模型 block_size 校验问题, 将 block_size 来源从全局 cache_config 改为 kv_cache_spec, 并通过新增的 _set_isa_to_layers 将 ISA 统一写回各注意力层。
- csrc/cpu/cpu_fused_moe.cpp (模块 MoE 内核; 类别 source; 类型 core-logic; 符号 fused_moe_impl): 修复 fused MoE scratch buffer 尺寸计算在 int32_t 下溢出导致 aligned_alloc 崩溃的核心问题, 是本 PR 最严重的 bug 修复。
- vllm/model_executor/layers/mamba/gdn/qwen_gdn_linear_attn.py (模块 GDN 模型层; 类别 source; 类型 data-contract; 符号 _log_gdn_backend_decision): CPU 平台

GDN 路径的内核选择与日志适配，避免 CPU 上误触发 flashinfer/CuteDSL 相关逻辑并产生噪音日志。

- vllm/triton_utils/importing.py (模块 Triton 导入; 类别 source; 类型 core-logic; 符号 HAS_TRITON) : 调整 Triton 可用性检查顺序, CPU-only 平台将 CPU driver 视为活跃驱动, 抑制误导性警告并避免 Triton 被误禁用。
- vllm/model_executor/layers/utils.py (模块 GEMM 分派; 类别 source; 类型 data-contract; 符号 dispatch_cpu_unquantized_gemm) : 修复小 GEMM 路径下 weight_packed_linear 的调用契约 (预打包 + 预分配输出), 并就地转换 bias 为 float32, 直接对应 Qwen3.6 router GEMM 问题。
- csrc/cpu/sgl-kernels/gemm.cpp (模块 GEMM 内核; 类别 source; 类型 core-logic; 符号 weight_packed_linear) : weight_packed_linear 从返回 Tensor 改为写入预分配 out, 简化内存管理并支持任意前导维度输出, 是 GEMM 契约重构的内核侧落地。
- csrc/cpu/torch_bindings.cpp (模块 算子绑定; 类别 source; 类型 core-logic; 符号 weight_packed_linear) : 同步更新 weight_packed_linear 的 TORCH_LIBRARY 绑定 schema, 匹配 C++ 新签名, 保证算子注册一致。
- tests/conftest.py (模块 测试夹具; 类别 test; 类型 test-coverage; 符号 _default_block_size) : 按平台差异化默认 block_size, CPU 上使用 128 以满足 SGL 内核 MultipleOf(32) 要求, 同时保证 XPU/ 其他平台行为不变。
- tests/v1/kv_connector/unit/test_bidirectional_kv_transfer.py (模块 KV 传输测试; 类别 test; 类型 test-coverage) : block_size 从 16 改为 32, 适配 CPU SGL 内核的块大小约束, 避免 CPU 上 KV connector 测试失败。
- tests/v1/attention/test_group_head_counts.py (模块 注意力测试; 类别 test; 类型 test-coverage) : 小范围调整测试中的 block_size 相关配置, 适配 CPU 注意力后端新的块大小约束。

关键符号: weight_packed_linear_cpu, _set_isa_to_layers, _log_gdn_backend_decision, dispatch_cpu_unquantized_gemm, weight_packed_linear, _default_block_size

关键源码片段

vllm/_custom_ops.py

新增 weight_packed_linear_cpu 包装函数, 将 C++ 端 weight_packed_linear 的「预分配输出」契约封装为与调用方匹配的 Python 接口, 同时移除不再匹配新签名的 register_fake 实现。

```
def weight_packed_linear_cpu(
    x: torch.Tensor,
    packed_weight: torch.Tensor,
    n: int,
    bias: torch.Tensor | None,
) -> torch.Tensor:
    # 先按输出维度 n 分配结果张量, 形状与输入 x 的前导维度保持一致。
    # 底层 C++ op 新签名是 void(out, mat1, mat2, bias, is_vnni) ——
    # 输出必须由调用方预分配, 这里负责 Tensor 创建与展平。
    output = torch.empty((*x.shape[0:-1], n), dtype=x.dtype)
```

```

# 任意形状输入展平为 [M, K] 交给 SGL kernel;
# is_vnni=True 表示 packed_weight 已由 convert_weight_packed 预打包,
# C++ 端不再做格式转换, 只为小 GEMM 保留低开销路径。
torch.ops._C.weight_packed_linear(
    output.reshape(-1, n),
    x.reshape(-1, x.size(-1)),
    packed_weight,
    bias,
    True,
)
return output

```

vllm/v1/attention/backends/cpu_attn.py

修复纯 SWA 模型 block_size 校验问题, 将 block_size 来源从全局 cache_config 改为 kv_cache_spec, 并通过新增的 _set_isa_to_layers 将 ISA 统一写回各注意力层。

```

class CPUAttentionMetadataBuilder(AttentionMetadataBuilder[CPUAttentionMetadata]):
    def __init__(
        self,
        kv_cache_spec: AttentionSpec,
        layer_names: list[str],
        vllm_config: VllmConfig,
        device: torch.device,
    ) -> None:
        # ... 其他初始化逻辑 ...
        # 关键修复: block_size 必须取自 kv_cache_spec 而非全局 cache_config。
        # 纯 SWA 模型的注意力组可能使用与全局不同的块大小, 原来读全局配置
        # 会导致 ISA 基于错误的 block_size 计算, 进而触发内核块大小校验失败。
        self.block_size = kv_cache_spec.block_size
        self.kv_cache_dtype = vllm_config.cache_config.cache_dtype
        self.isa = _get_attn_isa(
            self.dtype,
            self.block_size,
            self.head_dim,
            self.kv_cache_dtype,
        )
        # 统一把 ISA 写回本组所有注意力层; forward/do_kv_cache_update
        # 改读 layer.isa, 避免各层在 impl __init__ 中自行推算导致的时序不一致
        self._set_isa_to_layers(layer_names)

    def _set_isa_to_layers(self, layer_names: list[str]) -> None:
        attn_layers = get_layers_from_vllm_config(
            self.vllm_config,
            Attention,
            layer_names,
        )
        for layer in attn_layers.values():
            layer.isa = self.isa # type: ignore

```

csrc/cpu/cpu_fused_moe.cpp

修复 fused MoE scratch buffer 尺寸计算在 int32_t 下溢出导致 aligned_alloc 崩溃的核心问题，是本 PR 最严重的 bug 修复。

```
// 以下所有缓冲区大小 / 偏移都是跨整个 (token_num * topk_num) batch 的字节数。
// 对 Qwen3-235B-A22B 这类大模型在 TP4、max_num_batched_tokens=16384 下，
// 单个缓冲区（如 w2_gemm_output）字节数可达  $2^{31}$ ，超出 INT32_MAX。
// 若用 int32_t 累加，赋值时截断为负值，随后符号扩展成  $\sim 2^{64}$  的
// 巨大字节数传给 ScratchPadManager::realloc(size_t)，直接触发
// aligned_alloc 失败——表现像 " 分配失败 "，根因其实是溢出。
int64_t common_buffer_offset = 0;
int64_t w13_thread_buffer_offset = 0;
int64_t ws_thread_buffer_offset = 0;

// 公共缓冲区：token-expert 映射表 + 两段 GEMM 输出，全部按 64 字节对齐
const int64_t token_num_per_group_buffer_size =
    cpu_utils::round_up<64>(expert_num * sizeof(int32_t));
const int64_t token_num_per_group_buffer_offset = common_buffer_offset;
common_buffer_offset += token_num_per_group_buffer_size;

// w2 GEMM 输出缓冲直接依赖 token_num * topk_num，
// 是本次溢出最可能触发的路径；显式 static_cast<int64_t> 提升乘法
const int64_t w2_gemm_output_buffer_size =
    cpu_utils::round_up<64>(static_cast<int64_t>(token_num) * topk_num *
        output_size_2 * sizeof(float));
const int64_t w2_gemm_output_buffer_offset = common_buffer_offset;
common_buffer_offset += w2_gemm_output_buffer_size;

// 最终总大小 = 公共缓冲区 + 每线程缓冲 * 线程数，汇总后一次性 realloc
const int64_t buffer_size =
    common_buffer_offset +
    std::max(w13_thread_buffer_offset, ws_thread_buffer_offset) * thread_num;
cpu_utils::ScratchPadManager::get_scratchpad_manager()->realloc(buffer_size);
```

评论区精华

本 PR 的 review 过程非常简短：claude[bot] 说明这是 fork PR、自动审核被禁用；维护者 jikunshang 直接批准 (APPROVED)，没有留下技术评论或未解决疑虑。有价值的技术权衡都沉淀在提交历史中：c27e8c2 (int32 溢出修复) 详细解释了「32 位截断 → 符号扩展为 $\sim 2^{64}$ → aligned_alloc 失败」的根因链路，是典型的崩溃表象掩盖数值溢出案例；10979d2 (fix small gemm size) 与 5025b3f (fix) 共同演进出了 weight_packed_linear 的新契约——「调用方预打包 + 预分配输出」，说明该签名变更经过了反复调整；提交 5b2f083 (fix rocm) 表明 PR 在合入前还适配了 ROCm 平台的检查逻辑，体现跨平台回归意识。

- 暂无高价值评论线程

风险与影响

- 风险:

1. C++ ABI 签名变更 (高风险): `weight_packed_linear` 由「返回 Tensor」改为「void + out 参数」, 任何未同步更新的直接调用方都会在运行时失败。当前调用面已收敛到 `_custom_ops.weight_packed_linear_cpu`, 但需确认仓库内无其他路径直接调用 `torch.ops._C.weight_packed_linear`。
2. 移除 `register_fake` (中风险): `weight_packed_linear_fake` 删除后, 该 op 在 `torch.compile` 追踪下缺少 meta 实现; CPU 后端目前以 `eager` 为主, 但后续启用编译路径可能报 `missing fake kernel`。
3. `layer.bias` 就地 dtype 转换 (中风险): `dispatch_cpu_unquantized_gemm` 直接把 `layer.bias.data` 原地转 `float32`, 而不是创建临时副本; 若 `remove_weight=False` 且后续模块按原 dtype 读取 `bias`, 可能观察到 dtype 漂移。
4. 默认块大小上调 (低风险): CPU 测试默认 `block_size` 从 16 提到 128, KV cache 内存占用上升约 8 倍, CPU CI 资源消耗可能增加; 但更贴近生产配置。
5. TORCH_CHECK(is_vnni) 收紧 (行为破坏): C++ 端强制预打包权重, 绕过 Python 封装直接传未打包权重会直接报错而非静默转换。- 影响: 用户侧: CPU 上运行 MoE 大模型 (Qwen3-235B-A22B 等) 不再遭遇 `scratch buffer` 分配崩溃; Qwen3.6 等模型 `router logits` 计算恢复正确; 纯 SWA 模型可在 CPU 后端正常初始化与推理。系统侧: `weight_packed_linear` 契约重构让小型 GEMM 路径开销更低 (预打包一次、多次复用), `bias` 转 `float32` 提升数值精度; ISA 统一传播消除了多层注意力组之间的 ISA 不一致隐患。团队侧: CPU 测试默认配置向生产对齐, 后续 CPU 相关 CI 覆盖面更接近真实场景, 但测试内存占用上升, 需关注 CPU CI 资源。影响程度: 全部变更限定在 CPU 平台路径与相关测试, GPU/XPU 行为不受影响 (仅 `_default_block_size` 对 XPU 保持 64、其他平台 16 不变)。- 风险标记: C++ ABI 签名变更, `int32` 溢出修复 (核心内核路径), 移除 `register_fake` 影响编译路径, `bias` dtype 就地转换副作用, 默认 `block_size` 上调影响内存

关联脉络

- PR #52191 [CPU] Support FP16/BF16 persisted GDN state on AMX: 同改 `qwen_gdn_linear_attn.py` 的 CPU GDN 路径, 本 PR 补全了该文件在 CPU 平台上的内核选择与日志分支。
- PR #53536 [XPU] Ensure unquantized linear weight is N-contiguous: 同属 CPU/XPU 平台线性层 `dispatch` 优化, 与 `layers/utils.py` 的 GEMM 分派修改处于同一功能线。
- PR #52912 [Bugfix][KVOffload] P2P tier declares REQUEST_LEVEL on the producer leg: KV offload 系列测试与本 PR 的 `block_size` 调整联动, CPU 块大小约束变化会影响相关 `connector` 测试配置。
- PR #54299 [Perf] Avoid h2d copies from non-pinned CPU tensors: 同属近期 CPU 平台正确性与性能修复集群, 体现了 CPU 后端的密集稳定化阶段。