PR #41300 完整报告

vllm-project/vllm

[DeepSeek] Use torch.mm for bf16xbf16->fp32 gemm

合并时间：2026-05-01 07:28

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/41300

执行摘要

- 一句话：删除自定义 bf16→fp32 GEMM，改用 torch.mm
- 推荐动作：该 PR 是清理自定义算子的好示例，展示了如何利用 PyTorch 原生功能替代手写 CUDA 扩展。对于希望减少自定义代码依赖的开发者有参考价值。建议验证环境中的 PyTorch 版本是否支持 torch.mm(..., out_dtype=...)。整体风险可控，可合入。

功能与动机

发现 torch.mm 原生支持 bf16×bf16→fp32 矩阵乘法（底层调用 nvjet_sm100_tss_16x64_64x16_4x1_v_bz_TNN），无需再通过自定义 cuBLAS 包装实现，从而简化代码并减少维护成本。PR body 中通过脚本验证了该行为。

实现拆解

1. 删除自定义 Python 算子：在 vllm/_custom_ops.py 中移除 router_gemm_bf16_fp32 函数及其 fake 注册 router_gemm_bf16_fp32_fake。
2. 删除中间包装函数：在 vllm/model_executor/layers/utils.py 中删除 cublas_gemm_bf16_bf16_fp32，该函数仅委托给 ops.router_gemm_bf16_fp32。
3. 替换调用点：在 vllm/model_executor/layers/deepseek_v4_attention.py 的 attn_gemm_parallel_execute 中，将 cublas_gemm_bf16_bf16_fp32 替换为 torch.mm 并指定 out_dtype=torch.float32；在 vllm/model_executor/layers/fused_moe/router/gate_linear.py 的 forward Tier 2 中，将 ops.router_gemm_bf16_fp32 替换为 torch.mm。
4. 删除 C++ 实现：移除 csrc/moe/router_gemm.cu（自定义 cuBLAS GEMM 实现）、csrc/moe/moe_ops.h 中的声明以及 csrc/moe/torch_bindings.cpp 中的绑定，并从 CMakeLists.txt 中移除对应的编译源。
5. 无测试配套变更：本次改动未添加或修改测试，依赖现有测试覆盖。

关键文件：

- vllm/_custom_ops.py（模块 自定义算子；类别 source；类型 core-logic；符号 router_gemm_bf16_fp32, router_gemm_bf16_fp32_fake）：删除了自定义算子 router_gemm_bf16_fp32 及其 fake 注册，是本次重构的核心入口。
- vllm/model_executor/layers/utils.py（模块 工具层；类别 source；类型 data-contract；符号 cublas_gemm_bf16_bf16_fp32）：删除了 cublas_gemm_bf16_bf16_fp32 包装函数，该函数只委托给自定义算子。

- vllm/model_executor/layers/deepseek_v4_attention.py（模块 注意力层；类别 source；类型 data-contract）：两个关键调用点从 cublas_gemm_bf16_bf16_fp32 改为 torch.mm，影响注意力压缩器和索引器路径。
- csrc/moe/router_gemm.cu（模块 路由内核；类别 other；类型 deletion）：删除了整个自定义 cuBLAS GEMM 内核实现，是物理删除的核心。
- vllm/model_executor/layers/fused_moe/router/gate_linear.py（模块 门控线性层；类别 source；类型 data-contract）：在 MoE 路由前向传播的 Tier 2 中替换了 GEMM 调用，影响路由逻辑。

关键符号：router_gemm_bf16_fp32, router_gemm_bf16_fp32_fake, cublas_gemm_bf16_bf16_fp32, attn_gemm_parallel_execute, GateLinear.forward

关键源码片段

vllm/model_executor/layers/deepseek_v4_attention.py

两个关键调用点从 cublas_gemm_bf16_bf16_fp32 改为 torch.mm，影响注意力压缩器和索引器路径。

```
def attn_gemm_parallel_execute(self, hidden_states) -> tuple[Any, ...]:
    assert self.aux_stream_list is not None
    assert len(self.aux_stream_list) >= 3

    # fused_wqa_wkv (heaviest) on default; the three lighter input GEMMs
    # on aux streams 0..2 when their owning module exists.
    aux_fns: list[Callable[[], Any] | None] = [None, None, None]

    if self.compressor is not None:
        compressor = self.compressor

        def compressor_kv_score() -> torch.Tensor:
            # 使用 torch.mm 实现 bf16×bf16→fp32 GEMM
            return torch.mm(
                hidden_states,
                compressor.fused_wkv_wgate.weight.T,
                out_dtype=torch.float32,
            )

        aux_fns[0] = compressor_kv_score

    if self.indexer is not None:
        indexer = self.indexer

        def indexer_weights_proj() -> torch.Tensor:
            weights, _ = indexer.weights_proj(hidden_states)
            return weights

        def indexer_compressor_kv_score() -> torch.Tensor:
            return torch.mm(
```

```
                hidden_states,
                indexer.compressor.fused_wkv_wgate.weight.T,
                out_dtype=torch.float32,
            )

        aux_fns[1] = indexer_weights_proj
        aux_fns[2] = indexer_compressor_kv_score

    def fused_wqa_wkv() -> torch.Tensor:
        qr_kv, _ = self.fused_wqa_wkv(hidden_states)
        return qr_kv

    qr_kv, (kv_score, indexer_weights, indexer_kv_score) = execute_in_parallel(
        fused_wqa_wkv,
        aux_fns,
        self.ln_events[0],
        self.ln_events[1:4],
        self.aux_stream_list[:3],
    )

    return qr_kv, kv_score, indexer_kv_score, indexer_weights
```

vllm/model_executor/layers/fused_moe/router/gate_linear.py

在 MoE 路由前向传播的 Tier 2 中替换了 GEMM 调用，影响路由逻辑。

```
  def forward(self, x: torch.Tensor) -> torch.Tensor | tuple[torch.Tensor, Parameter | None]:
    import vllm._custom_ops as ops

    # Tier 1: DSV3 specialized kernel
    if self.allow_dsv3_router_gemm and x.shape[0] <= 16:
        output = ops.dsv3_router_gemm(
            hidden_states=x,
            router_weight=self.weight,
            output_dtype=self.out_dtype,
        )
        return output, None

    # Tier 2: cuBLAS bf16→fp32 (now using native torch.mm with out_dtype)
    if self.allow_cublas_router_gemm and x.dtype == torch.bfloat16:
        output = torch.mm(x, self.weight.T, out_dtype=torch.float32)
        return output, None

    # Tier 3: F.linear (ReplicatedLinear)
    if self.out_dtype is not None and x.dtype != self.weight.dtype:
        x = x.to(self.weight.dtype)
    output, output_bias = super().forward(x)
    if self.out_dtype is not None and output.dtype != self.out_dtype:
        output = output.to(self.out_dtype)
    return output, output_bias
```

评论区精华

gemini-code-assist[bot] 对 torch.mm 的 out_dtype 参数提出了兼容性质疑，指出该参数并非标准 PyTorch 2.5 及之前版本的公共 API，可能导致 TypeError；同时提醒 torch.mm 仅支持 2D 输入，若 hidden_states 为 3D 将引发运行时错误。然而，该 PR 最终被 tlrmchlsmth 批准，表明团队内部已确认所需 PyTorch 版本的兼容性（可能依赖 PyTorch 2.6+），或该路径中的输入形状已保证为 2D。

- torch.mm out_dtype 参数兼容性及输入维度要求 (correctness): PR 被 tlrmchlsmth 批准，暗示团队已确认环境满足 PyTorch 版本要求且输入形状为 2D，或后续将处理兼容性。

风险与影响

- 风险：
  1. PyTorch 版本兼容性：torch.mm 的 out_dtype 参数在 PyTorch 2.5 及以下版本不存在，使用旧版 PyTorch 的用户会遇到 TypeError。vLLM 通常依赖较新版本的 PyTorch（≥2.6），但若实际环境中版本不满足，需添加 fallback 或版本检查。
  2. 输入形状假设：torch.mm 严格要求 2D 输入，若 hidden_states 在推理时可能以 3D shape（如 [batch, seq, hidden]）传入，则会失败。当前调用点处于合并了 batch 和 seq 维度的上下文中（注意力层输入已铺平），风险较低，但仍需确认。
  3. 回归风险：删除自定义 cuBLAS 包装可能影响性能或精度，但 PyTorch 原生实现应同样基于 cuBLAS，且 out_dtype 保证了 fp32 精度，回归可能性较小。 - 影响：对用户：仅影响 DeepSeek V4/V3 模型的推理路径，需使用支持 out_dtype 的 PyTorch 版本。对开发团队：消除了一个自定义 cuBLAS 包装，降低了 C++ 代码的编译和维护负担。对系统：编译时间略有减少，二进制体积缩小。 - 风险标记：PyTorch out_dtype 兼容性，输入 2D 维度假设，无测试变更

关联脉络

- PR #41419 Fix typo in log message for indexer cache: 修改了相同文件 deepseek_v4_attention.py，属于同模块的修复。
- PR #41374 [DSV4] Avoid redundant dtype conversion.: 同属 DeepSeek V4 优化系列，本 PR 进一步简化了 GEMM 实现。