

PR #53838 完整报告

vllm-project/vllm

[ROCm][DSV4][Perf] Fuse DeepSeek V4 C4 compressor GEMMs

合并时间: 2026-08-26 23:36

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/53838>

执行摘要

- 一句话: 融合 DSV4 C4 压缩器 GEMM, ROCm 解码提速约 1.5 倍
- 推荐动作: 值得精读, 尤其关注三点: 一是 `prepare_compressor_gemm_fusion` 中 `set_` + 非持久 `buffer` 的零拷贝权重打包手法, 既保留参数身份和 `checkpoint` 键, 又不产生重复存储; 二是 `offloader` 兼容性防线 (`NoopOffloader` 检查) 设计得很干净, 是性能优化与其它子系统共存的好范例; 三是测试对存储别名不变量 (`data_ptr`、`storage_offset`、`state_dict` 键) 的精确断言, 值得在类似融合优化中复制。整体改动小而聚焦, 约 1.4~1.5x 的 GEMM 加速和 2~3% 的端到端收益均有完整实测支撑。

功能与动机

PR body 说明: DeepSeek V4 C4 target-model 层的主压缩器投影与 `indexer` 压缩器投影使用相同的输入和 K 维度, 但 ROCm 路径上这两个 GEMM 串行执行, kernel 效率低。作者的目标是把兼容的 GEMM 合并到现有单流上执行, 不改 graph 或 stream 调度。实测基准 (gfx950/HIP 7.2.53211, M=4/K=7168/N=2048+512): `eager` 中位数 42.04us -> 29.16us (1.442x), `graph-replay` 37.16us -> 24.44us (1.520x); TP8 服务下 TPOT 改善 2.15%~2.81%, 吞吐提升 2.15%~2.85%, 且输入输出 token 总数与 baseline 完全一致。

实现拆解

本 PR 的变更入口是权重加载完成后的 `process_weights_after_loading` 钩子, 核心改动集中在 ROCm 专属的 `DeepseekV4ROCMAttnMLAAttention`, 实现拆解如下:

1. 新增融合状态 (`vllm/models/deepseek_v4/amd/rocm.py`): 在 `DeepseekV4ROCMAttnMLAAttention.__init__` 中注册非持久 `buffer` `_fused_compressor_weight` (`persistent=False`, 避免进入 `state_dict` 与 `checkpoint` 键冲突), 并新增 `_fused_compressor_split_sizes: tuple[int, int] | None` 记录主压缩器与 `indexer` 的输出列数切分。
2. 权重打包入口 `prepare_compressor_gemm_fusion` (`rocm.py`): 幂等设计, 已融合直接返回 `False`; 先检查 `get_offloader()` 是否为 `NoopOffloader`, `offload` 激活时告警并跳过 (`offloader` 会重绑定子参数、破坏 `set_` 建立的存储别名); 随后校验两个权重同为 2D、共享 K 维度、`dtype` 与 `device`; 满足条件后沿 `dim 0` 一次性 `torch.cat` 拼接为 [2560, 7168], 并在 `torch.no_grad()` 下用 `set_` 把两个原参数原地重绑定为拼接 `buffer` 的切片视图, 实现零拷贝别名, 同时保留参数身份与加载后的 `checkpoint` 键。

3. 前向融合路径 `_run_parallel_input_projections` (rocm.py) : 融合生效时用一次 `torch.mm(hidden_states, fused_weight.t(), out_dtype=torch.float32)` (即 $[M, 7168] \times [7168, 2560]$) 替代原来的两次串行 mm, 产出按 `_fused_compressor_split_sizes` 切分为零拷贝视图交给现有消费方; 未融合时回退到原双 GEMM 路径, 保证兼容性。
4. 加载后处理钩子 (vllm/models/deepseek_v4/amd/model.py) :
`DeepseekV4ForCausalLM.process_weights_after_loading` 在遍历模块时先调用 `prepare_compressor_gemm_fusion()` (在 `prepare_attn_preshuffle` 之前, 因为 `preshuffle` 只处理注意力相关权重、不接触压缩器权重), 累计融合层数并打一条 `logger.info` 日志。
5. 测试配套 (tests/models/test_deepseek_v4_rocm_compressor_gemm_fusion.py, 新增 141 行) : 3 个单元测试分别覆盖: 融合后两个原参数与拼接 buffer 共享同一 storage (用 `untyped_storage().data_ptr()` 与 `storage_offset()` 精确断言) 且 `state_dict` 键保持不变、重复调用幂等; offload 场景下跳过融合; 融合路径只触发一次 `torch.mm` 调用且输出元组语义不变。测试通过 `__new__` 手工构造最小 attention 实例, 避免初始化大模型。

关键文件:

- vllm/models/deepseek_v4/amd/rocm.py (模块 ROCM 层; 类别 source; 类型 core-logic ; 符号 `prepare_compressor_gemm_fusion`, `_run_parallel_input_projections`, `init`) : 核心实现文件: 新增非持久 buffer 状态、`prepare_compressor_gemm_fusion` 权重打包入口、`_run_parallel_input_projections` 单 GEMM 融合前向路径, 是本次性能优化的主体。
- vllm/models/deepseek_v4/amd/model.py (模块 模型加载; 类别 source; 类型 core-logic ; 符号 `process_weights_after_loading`) : 将融合入口挂接到权重加载后处理流程: 在 `process_weights_after_loading` 中逐层调用 `prepare_compressor_gemm_fusion`, 并统计融合层数打日志。
- tests/models/test_deepseek_v4_rocm_compressor_gemm_fusion.py (模块 单元测试; 类别 test; 类型 test-coverage; 符号 `_WeightOnlyLinear`, `_Compressor`, `_IndexerWeightsProjection`, `_Indexer`) : 新增 3 个单元测试, 精确断言存储别名不变量 (`data_ptr`、`storage_offset`)、`state_dict` 兼容性、offload 跳过逻辑与单次 mm 调用行为, 是融合正确性的主要保障。

关键符号: `prepare_compressor_gemm_fusion`, `_run_parallel_input_projections`, `process_weights_after_loading`

关键源码片段

vllm/models/deepseek_v4/amd/rocm.py

核心实现文件: 新增非持久 buffer 状态、`prepare_compressor_gemm_fusion` 权重打包入口、`_run_parallel_input_projections` 单 GEMM 融合前向路径, 是本次性能优化的主体。

```
# 以下代码位于 vllm/models/deepseek_v4/amd/rocm.py 的 DeepseekV4ROCMAttnMLAAttention。
# __init__ 中新增的融合状态:
# self._fused_compressor_weight: torch.Tensor | None
# 注册为非持久 buffer, 避免进入 state_dict, 保证 checkpoint 键不变:
# self.register_buffer("_fused_compressor_weight", None, persistent=False)
# self._fused_compressor_split_sizes: tuple[int, int] | None = None
```

```

def prepare_compressor_gemm_fusion(self) -> bool:
    # 幂等：已融合过的层直接返回 False，不重复打包、不重建 buffer
    if self._fused_compressor_weight is not None:
        return False

    from vllm.model_executor.offloader import NoopOffloader, get_offloader

    # offloader 会重绑定子参数并破坏基于 set_ 的存储别名，
    # 因此权重 offload 激活时跳过融合，保持与 offload 兼容
    if not isinstance(get_offloader(), NoopOffloader):
        logger.warning_once(
            "DeepSeek V4 compressor GEMM fusion is incompatible with "
            "weight offloading and will remain disabled."
        )
        return False

    compressor = self.compressor
    indexer = self.indexer
    if compressor is None or indexer is None:
        return False

    main_weight = compressor.fused_wkv_wgate.weight
    indexer_weight = indexer.compressor.fused_wkv_wgate.weight
    # 两个投影权重必须同为 2D 矩阵，且共享 K 维度、dtype 与 device
    if main_weight.ndim != 2 or indexer_weight.ndim != 2:
        raise ValueError("DeepSeek V4 compressor weights must be matrices")
    if main_weight.shape[1] != indexer_weight.shape[1]:
        raise ValueError("DeepSeek V4 compressor weights must share K")
    if main_weight.dtype != indexer_weight.dtype:
        raise ValueError("DeepSeek V4 compressor weights must share dtype")
    if main_weight.device != indexer_weight.device:
        raise ValueError("DeepSeek V4 compressor weights must share device")

    main_size = main_weight.shape[0]
    indexer_size = indexer_weight.shape[0]
    # 加载后只做一次拼接：[2048, 7168] + [512, 7168] -> [2560, 7168]
    fused_weight = torch.cat((main_weight, indexer_weight), dim=0)
    with torch.no_grad():
        # 原地把两个原参数重绑定为拼接 buffer 的切片视图（零拷贝），
        # 保留参数身份与 checkpoint 键，不保留重复存储
        main_weight.set_(fused_weight[:main_size])
        indexer_weight.set_(fused_weight[main_size:])

    self._fused_compressor_weight = fused_weight
    self._fused_compressor_split_sizes = (main_size, indexer_size)
    return True

```

[vllm/models/deepseek_v4/amd/model.py](#)

将融合入口挂接到权重加载后处理流程：在 `process_weights_after_loading` 中逐层调用 `prepare_compressor_gemm_fusion`，并统计融合层数打日志。

```
# vllm/models/deepseek_v4/amd/model.py 中 DeepseekV4ForCausalLM 的权重后处理钩子。  
# 该钩子在 per-layer 量化 finalize 之后、图捕获之前执行，  
# 因此先做 GEMM 融合、再做注意力权重 preshuffle 是安全的：  
# preshuffle 只处理 fused_wqa_wkv 与 wo_b，不接触压缩器权重。
```

```
def process_weights_after_loading(self) -> None:  
    # After per-layer quant finalize, so we preshuffle the final fp8 weights.  
    fused_compressor_layers = 0  
    for module in self.modules():  
        if isinstance(module, DeepseekV4ROCMAttnMLAAttention):  
            fused_compressor_layers += module.prepare_compressor_gemm_fusion()  
            module.prepare_attn_preshuffle()  
        elif isinstance(module, DeepseekV4MLP):  
            module.prepare_gateup_preshuffle()  
    if fused_compressor_layers:  
        logger.info(  
            "Fused the C4 compressor GEMMs in %d DeepSeek V4 layers",  
            fused_compressor_layers,  
        )
```

tests/models/test_deepseek_v4_rocm_compressor_gemm_fusion.py

新增 3 个单元测试，精确断言存储别名不变量（`data_ptr`、`storage_offset`）、`state_dict` 兼容性、`offload` 跳过逻辑与单次 `mm` 调用行为，是融合正确性的主要保障。

```
# tests/models/test_deepseek_v4_rocm_compressor_gemm_fusion.py
```

```
def test_prepare_compressor_gemm_fusion_aliases_original_weights():  
    # 主压缩器权重 4x3，indexer 权重 2x3，共享 K=3  
    main_value = torch.arange(12, dtype=torch.float32).reshape(4, 3)  
    indexer_value = torch.arange(6, dtype=torch.float32).reshape(2, 3) + 100  
    attention = _make_attention(main_value, indexer_value)  
    main_weight = attention.compressor.fused_wkv_wgate.weight  
    indexer_weight = attention.indexer.compressor.fused_wkv_wgate.weight  
  
    assert attention.prepare_compressor_gemm_fusion() is True  
  
    # 拼接 buffer 内容正确，切分尺寸为 (4, 2)  
    fused_weight = attention._fused_compressor_weight  
    assert fused_weight is not None  
    assert attention._fused_compressor_split_sizes == (4, 2)  
    torch.testing.assert_close(fused_weight[:4], main_value)  
    torch.testing.assert_close(fused_weight[4:], indexer_value)  
  
    # 关键不变量：两个原参数与拼接 buffer 共享同一底层 storage（零拷贝别名）  
    fused_storage = fused_weight.untyped_storage().data_ptr()
```

```

assert main_weight.untyped_storage().data_ptr() == fused_storage
assert indexer_weight.untyped_storage().data_ptr() == fused_storage
assert main_weight.storage_offset() == 0
assert indexer_weight.storage_offset() == main_weight.numel()

# 非持久 buffer 不进入 state_dict, checkpoint 键保持原样
state_dict = attention.state_dict()
assert "_fused_compressor_weight" not in state_dict
torch.testing.assert_close(
    state_dict["compressor.fused_wkv_wgate.weight"], main_value
)
torch.testing.assert_close(
    state_dict["indexer.compressor.fused_wkv_wgate.weight"],
    indexer_value,
)

# 重复调用幂等：不会重建 buffer, data_ptr 保持不变
fused_data_ptr = fused_weight.data_ptr()
assert attention.prepare_compressor_gemm_fusion() is False
assert attention._fused_compressor_weight.data_ptr() == fused_data_ptr

```

评论区精华

本 PR 没有实质性的 review 评论线程。claude[bot] 提示 fork 来源的 PR 默认关闭自动化 review，可由维护者输入 [@claude review](#) 触发；维护者 shen-shanshan 在 CI 通过后直接批准 (LGTM)。作者在 PR body 中主动澄清了与既有实现的差异：确认没有重复实现（live 搜索 52911 与 "DeepSeek V4 ROCm compressor GEMM fusion" 无匹配）；#51794 是在辅助 HIP stream 上分别调度两个压缩器 GEMM，而本 PR 改为在现有单流上合并成一个 GEMM，不改变 graph 或 stream 调度；#53182 只更新 ROCm profiler runtime，不改变执行语义。设计上最关键的取舍是：为保持与权重 offload 的兼容性（offloader 可能重绑定子参数破坏存储别名），融合在 offload 激活时显式禁用，并用非持久 buffer 保证 checkpoint 键不变。

- 与既有 ROCm 实现的差异化说明 (design): 差异化明确，未引发争议；维护者直接 LGTM 合入。
- PR 评审与合并 (other): 无技术争议，评审通过并合入。

风险与影响

- 风险：
 1. 存储重绑定风险 (vllm/models/deepseek_v4/amd/rocm.py)：set_ 是就地修改 parameter 的 storage 元数据，任何在融合之后仍持有原 storage 引用、或依赖独立 storage 的组件（offloader、权重保存 / 重分片、LoRA 等）都可能受影响。目前仅对 offloader 做了显式防护（非 NoopOffloader 即跳过），若未来压缩器权重支持 fp8 量化或其他重绑定路径，需要同步补充兼容性检查。
 2. 数值精度差异：单次大 GEMM 的累加顺序与两次小 GEMM 不同，实测最大绝对差 $6.1035e-05$ ，通过 $rtol=atol=1e-4$ 阈值；但对更大 batch 或更长序列仍需持续观测。

3. CUDA graph 稳定性：融合发生在 `process_weights_after_loading`，早于 graph 捕获，`storage` 在捕获期稳定；但 `_fused_compressor_weight` 是非持久 buffer，若后续模型保存逻辑依赖 buffer 完整遍历，需要确保其缺失不报错（已有 `state_dict` 测试覆盖）。
4. 执行路径限定：融合只在 `DeepseekV4ROCMAIterMLAAttention`（ROCm 专属路径）生效，非 ROCm 平台、非 DSV4 模型不受影响；但校验条件只检查 `dtype/device/K` 一致，并未限定 BF16，若未来引入量化权重需确认融合后的单次 mm 仍走等价算子。- 影响：对用户：DeepSeek V4 系列在 ROCm（gfx950 + AITer 路线）后端默认启用融合，TP8 服务端到端吞吐提升 2.15%~2.85%，TPOT 改善 2.15%~2.81%，融合 GEMM 本身延迟下降 31%~34%；GSM8K 5-shot 严格 exact match 0.960576，精度不受影响。对系统：加载期额外一次权重拼接拷贝（总显存不变，[2048+512, 7168] BF16 约 36.8MB 合并为一块），前向少一次 kernel launch，decode 热路径收益明显；权重 offload 场景自动回退原路径。对团队：该 PR 提供了一个可复用的 "打包权重 + `set_` 零拷贝别名 + 非持久 buffer" 模式，后续 ROCm 侧类似 GEMM 融合可直接借鉴；缺点是把平台专属优化逻辑继续沉淀在 `amd/` 目录内，需要维护者注意与 offloader、量化等机制的联动。- 风险标记：`set_` 存储重绑定，offload 兼容性保护，数值精度差异，仅 ROCm/DSV4 生效

关联脉络

- PR #51794 DeepSeek V4 ROCm compressor GEMM scheduling on auxiliary HIP streams (PR body 提及)：PR body 明确对比的替代方案：保留两个压缩器 GEMM 并在辅助 HIP stream 上调度；本 PR 改为单流融合，二者是同一性能问题的不同解法。
- PR #53182 ROCm profiler runtime update (PR body 提及)：PR body 声明其只更新 ROCm profiler runtime、不改变执行语义，与本 PR 的 GEMM 融合无功能交集，属于同一 ROCm DSV4 性能优化线上相邻的改动。
- PR #53120 [Offloader] Offload submodules that make_layers never reaches: 该 PR 修复了多模态塔权重 CPU offload 静默失效的问题，触及 offloader 子参数重绑定逻辑；本 PR 的融合正是担心 offloader 重绑定破坏 `set_` 存储别名而显式禁用，两者在同一 offloader 机制上有直接交互。