

PR #55309 完整报告

vllm-project/vllm

[Qwen3.8-Flash-Next] Fuse PLE residual and QSA output gate

合并时间: 2026-09-14 20:16

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/55309>

执行摘要

- 一句话: Qwen4Exp 把 PLE 外层残差与 QSA 输出门并入 Triton 内核, 保持 BF16 舍入边界
- 推荐动作: 值得精读, 尤其是要维护 Qwen4Exp 或改造 Triton 内核的工程师。建议重点关注三处: (1) ops/ple.py::_ple_conv_kernel 中 (residual + conv_output).to(dtype) → FP32 加 outer_residual → store 的三步写法, 这是「融合但不改数值」的范式, 也是 test_ple.py 能用 torch.equal 逐位断言的原因; (2) commit 6d038df 的修复思路——把可选张量变为必填关键字参数以消除 specialization 分支、并同步 warmup 桩, 值得作为内核演进 checklist; (3) coderabbitai 关于 graph padding 行残差的两条未决意见 (ops/ple.py 与 test_qsa_reference.py), 建议在后续 PR 中补一个 spec + padding 混合批次的逐位用例来证实或推翻。

功能与动机

PR body 写明动机是去掉 Qwen4Exp NVIDIA 路径上的两个独立逐点算子: “This removes two standalone pointwise operations from the Qwen4Exp NVIDIA path”, 具体为 (1) fuse the outer PLE residual add into the existing short-convolution kernel, (2) fuse the optional QSA output gate into both the single-split attention epilogue and the split-K merge kernel。作者同时给出约束: “preserve the original BF16/FP16 rounding boundaries before each fused post-op”, 并把与 #55272 的关系写清楚: 本 PR 是互补而非重复, #55272 移除了 model-level compile/custom-op plumbing, 但其 diff 仍把 PLE 外层残差与 QSA gate 当作独立 PyTorch 运算执行, 若 #55272 先落地本 PR 需 rebase 到其 breakable-graph 调用结构。

实现拆解

1. 调用点: 模型侧去掉独立加法。vllm/models/qwen4_exp/nvidia/model.py 的 block forward 把 `hidden_states = hidden_states + self.ple(...)` 改为 `hidden_states = self.ple(...)`, 外层残差改由 PLE 层内部完成; ple_layer.py::Qwen4ExpPLELayer.forward 相应变成 `self._short_conv(conv_input, gated_output, hidden_states)`, 注释说明「short convolution accumulates into gated_output」。这样 PLE 的返回张量语义仍然是 `hidden_states + ple_out`, 对上层 block 完全透明。
2. 分发层透传新实参。`_short_conv_dilated_dispatch` 新增 `outer_residual` 形参, 按 `metadata.num_actual_tokens` 切片, 并在四条 `ple_conv` 调用 (spec、mixed batch 的

decode 段与 prefill 段、纯 decode) 上逐条透传; mixed batch 场景下与 `inputs/residual` 一样用 `torch.split([num_decode_tokens, num_prefill_tokens])` 拆成 `outer_residual_d/p`, 保证 logical row 与 token 索引映射顺序不变。

3. PLE 内核: 先舍入、再在 FP32 加外层残差。 `vllm/models/qwen4_exp/nvidia/ops/ple.py::_ple_conv_kernel` 新增 `outer_residual_ptr`, 把原来的 `tl.store(residual + conv_output)` 拆成三步: `(residual + conv_output).to(residual_ptr.dtype.element_ty)` 复现原 eager 的 BF16/FP16 单次舍入, 再在 FP32 中加 `outer_residual`, 最后 store 隐式截断回 BF16。`ple_conv` 宿主封装新增必填 `outer_residual`, docstring 更新为 “Add short convolution and the outer residual; update state。”
4. no-op 路径不再丢加法。`_short_conv` 在 profiling、`attn_metadata is None`、或该层 entry 缺失时, 从直接 `return` 改为先 `residual.add_(outer_residual)` 再返回——因为旧路径在这些场景下仍会执行 `hidden_states + self.ple(...)`; 这是「融合必须语义等价」原则的直接体现。
5. QSA 输出门融合。 `vllm/models/qwen4_exp/nvidia/qa.py` 的 `Qwen4ExpQSAAttention.forward` 删除 `flat_output = flat_output * torch.sigmoid(gate)`, 改为 `assert gate is not None` 并把 `gate` 传入 `_run_qsa → forward_qsa → qa_sparse_paged_attention(output_gate=output_gate[:num_tokens])`, `gate` 在 kernel 内乘完再交给 `o_proj`。
6. QSA 内核与 warmup 配套。 `ops/qa.py` 的 `_qa_sparse_paged_gqa_splitk_kernel` (`NUM_SPLITS == 1` 快路径) 与 `_qa_merge_splitk_kernel` 都新增 `output_gate_ptr` 及 `row/head stride`, 在 kernel 内做「先 `.to(output_ptr.dtype.element_ty)` 舍入、再 FP32 乘 `tl.sigmoid`」; `qa_sparse_paged_attention` 把 `output_gate` 改为必填关键字参数并 `assert output_gate.is_contiguous()`; `warmup_qa_sparse_paged_attention` 同步补上 `output_gate_ptr` 及两个 `stride`, 修复了 ZJY0516 在 commit 6d038df 中指出的位置参数错位。
7. 测试配套。 `tests/models/qwen4_exp/test_ple.py::test_fused_conv_correctness` 新增 `outer_residual` 随机张量, 把原来的 `assert_close(atol=rtol=3e-2)` 换成 `torch.equal` 逐位断言, 并对 `graph_padding` 行断言等于 `outer_residual + residual`;
`tests/models/qwen4_exp/test_qa_reference.py::test_qa_sparse_paged_attention_correctness` 新增 `output_gate = torch.randn_like(q)`, 参考结果改为 `expected * torch.sigmoid(output_gate)`。无配置项、schema 或部署变更。

性能数据 (PR body, B200 / CUDA 13.0 / BF16 / CUDA graphs 捕获): PLE 解码 1 行 6.464us → 4.480us (1.443x), 256 行仅 1.021x; 预填 16-8192 token 为 1.15x-1.29x。QSA 门控 1 行查询 12.832us → 11.616us (1.105x), 513 行只剩 1.011x。

关键文件:

- `vllm/models/qwen4_exp/nvidia/ops/ple.py` (模块 卷积内核; 类别 source; 类型 core-logic; 符号 `_ple_conv_kernel`, `ple_conv`) : PLE 短卷积 Triton 内核本体: 新增 `outer_residual_ptr` 并把「先舍入、再 FP32 加外层残差」写进 `_ple_conv_kernel`, 是本次数值等价融合的核心实现; `ple_conv` 封装同步新增必填参数。
- `vllm/models/qwen4_exp/nvidia/ops/qa.py` (模块 稀疏注意力; 类别 source; 类型 core-logic; 符号 `_qa_sparse_paged_gqa_splitk_kernel`, `_qa_merge_splitk_kernel`,

qsa_sparse_paged_attention, warmup_qsa_sparse_paged_attention) : QSA 稀疏分页注意力内核：单 split 快路径与 split-K merge 都新增 output_gate_ptr，并把 qsa_sparse_paged_attention 的 gate 改为必填关键字参数；warmup_qsa_sparse_paged_attention 的签名同步是维护者修复启动期 TypeError 的落点。

- vllm/models/qwen4_exp/nvidia/ple_layer.py (模块 短卷积层；类别 source；类型 data-contract；符号 _short_conv, _short_conv_dilated_dispatch, forward) : PLE 层分发与 no-op 语义：_short_conv_dilated_dispatch 新增并按 token 切片 / 透传 outer_residual, _short_conv 在 metadata 缺失时改为 residual.add_(outer_residual)，保证融合后与旧 hidden_states + self.ple(...) 完全等价。
- vllm/models/qwen4_exp/nvidia/qa.py (模块 注意力层；类别 source；类型 data-contract；符号 Qwen4ExpQSAFlashAttentionImpl.forward_qsa, Qwen4ExpQSAAttention._run_qsa, Qwen4ExpQSAAttention.forward) : QSA 注意力层接线：forward 删除独立的 torch.sigmoid(gate) 乘法并断言 gate 非空，_run_qsa/forward_qsa 把 gate 一路透传到内核，是「去掉一个逐点算子」的入口。
- vllm/models/qwen4_exp/nvidia/model.py (模块 模型主体；类别 source；类型 core-logic；符号 forward) : 模型 block 的调用点变更：hidden_states = hidden_states + self.ple(...) 变为 hidden_states = self.ple(...)，一行改动即完成外层残差的下沉，也是判断语义等价的关键位置。
- tests/models/qwen4_exp/test_ple.py (模块 卷积测试；类别 test；类型 test-coverage；符号 test_fused_conv_correctness) : PLE 融合正确性的主要防线：test_fused_conv_correctness 新增 outer_residual 随机张量，并把原来的 atol/rtol=3e-2 宽松比较升级为 torch.equal 逐位断言，还覆盖了 graph padding 行。
- tests/models/qwen4_exp/test_qsa_reference.py (模块 注意力测试；类别 test；类型 test-coverage；符号 test_qsa_sparse_paged_attention_correctness) : QSA 门控的参考测试：新增 output_gate 并把参考结果改为 expected * torch.sigmoid(output_gate)；但 coderabbitai 指出 2e-2 容差不足以校验 BF16 舍入顺序，该用例的强度仍是待改进项。

关键符号：_ple_conv_kernel, ple_conv, _short_conv, _short_conv_dilated_dispatch, Qwen4ExpQSAAttention.forward, Qwen4ExpQSAAttention._run_qsa, Qwen4ExpQSAFlashAttentionImpl.forward_qsa, qsa_sparse_paged_attention, _qsa_sparse_paged_gqa_splitk_kernel, _qsa_merge_splitk_kernel, warmup_qsa_sparse_paged_attention

关键源码片段

vllm/models/qwen4_exp/nvidia/ops/ple.py

PLE 短卷积 Triton 内核本体：新增 outer_residual_ptr 并把「先舍入、再 FP32 加外层残差」写进 _ple_conv_kernel，是本次数值等价融合的核心实现；ple_conv 封装同步新增必填参数。

ops/ple.py：内核收尾段与宿主封装（重新整理后的关键实现）

```
# _ple_conv_kernel 收尾段：短卷积结果先按原有 eager 边界舍入到 BF16/FP16，
# 再在 FP32 中加上 PLE 外层残差，最后一次性写回。
# 这样融合后的数值与 unfused 的 (hidden_states + ple_out) 逐位一致。
ple_output = (residual + conv_output).to(residual_ptr.dtype.element_ty)
```

```

outer_residual = tl.load(
    outer_residual_ptr + output_t * C + c_offs,
    mask=c_mask,
    other=0.0,
)
# 两个 BF16 值的和在 FP32 中可精确表示, store 时的隐式截断即为唯一一次舍入
ple_output = outer_residual.to(tl.float32) + ple_output.to(tl.float32)
if launch_pdl:
    tl.extra.cuda.gdc_launch_dependents()
tl.store(
    residual_ptr + output_t * C + c_offs,
    ple_output,
    mask=c_mask,
)

```

宿主侧封装: outer_residual 改为必填参数, decode/spec/prefill 三个 mode 都要传

```

def ple_conv(
    inputs: torch.Tensor,
    residual: torch.Tensor,
    conv_state: torch.Tensor,
    conv_weights: torch.Tensor,
    state_indices: torch.Tensor,
    outer_residual: torch.Tensor,
    *,
    mode: Literal["decode", "spec", "prefill"],
    dilation: int,
    query_start_loc: torch.Tensor | None = None,
    num_accepted_tokens: torch.Tensor | None = None,
    spec_query_len: int = 1,
    token_indices: torch.Tensor | None = None,
) -> None:
    """Add short convolution and the outer residual; update state."""

```

vllm/models/qwen4_exp/nvidia/ple_layer.py

PLE 层分发与 no-op 语义: `_short_conv_dilated_dispatch` 新增并按 token 切片 / 透传 `outer_residual`, `_short_conv` 在 metadata 缺失时改为 `residual.add_(outer_residual)`, 保证融合后与旧 `hidden_states + self.ple(...)` 完全等价。

`ple_layer.py`: 外层残差的分发、no-op 兜底与调用点

```

def _short_conv_dilated_dispatch(
    self,
    inputs: torch.Tensor,
    residual: torch.Tensor,
    outer_residual: torch.Tensor, # 新增: PLE 外层残差, 随 inputs/residual 一起裁剪
    metadata: PleShortConvAttentionMetadata,
    conv_state: torch.Tensor,
    conv_weights: torch.Tensor,

```

```

) -> None:
...
inputs = inputs[: metadata.num_actual_tokens]
residual = residual[: metadata.num_actual_tokens]
outer_residual = outer_residual[: metadata.num_actual_tokens]

if has_spec:
    # 混合批次保持原始 row 顺序, kernel 通过 token_indices 做逻辑行映射
    ple_conv(
        inputs=inputs,
        residual=residual,
        conv_state=conv_state,
        conv_weights=conv_weights,
        state_indices=spec_state_indices,
        outer_residual=outer_residual,
        mode="spec",
        ...
    )

if has_prefill:
    if non_spec_token_indices is None:
        # 与 inputs/residual 相同的切分方式, 保证行序一致
        outer_residual_d, outer_residual_p = torch.split(
            outer_residual, [num_decode_tokens, num_prefill_tokens], dim=0
        )
    else:
        outer_residual_d = outer_residual_p = outer_residual
    ...

# State routing 依赖运行时请求 metadata, 因此留在图外。
@eager_break_during_capture
def _short_conv(
    self,
    inputs: torch.Tensor,
    residual: torch.Tensor,
    outer_residual: torch.Tensor,
) -> None:
    forward_context = get_forward_context()
    attn_metadata = forward_context.attn_metadata
    # Profiling 会省略全部 metadata 或该 Mamba entry。此处短卷积是 no-op,
    # 但外层残差加法仍必须发生, 否则等价于丢掉 hidden_states。
    if attn_metadata is None:
        residual.add_(outer_residual)
        return
    ...
    layer_attn_metadata = attn_metadata.get(self.prefix)
    if layer_attn_metadata is None:
        residual.add_(outer_residual)

```

```

        return
    ...
    self._short_conv_dilated_dispatch(
        inputs=inputs,
        residual=residual,
        outer_residual=outer_residual,
        metadata=layer_attn_metadata,
        conv_state=conv_state,
        conv_weights=conv_weights.to(dtype=inputs.dtype),
    )

```

调用点：短卷积把外层残差累加进 gated_output，模型侧不再单独做加法

```

self._short_conv(conv_input, gated_output, hidden_states)
return gated_output

```

评论区精华

- 维护者 ZJY0516 直接以 commit 6d038df 修正启动期崩溃，commit message 写得很明确：“warmup_qsa_sparse_paged_attention still used the pre-fusion kernel signatures, so every .warmup() call misaligned positionally and raised TypeError at startup with enable_jit_warmup=True (the default). Pass the gate tensor and its strides in both warmup calls. attn_output_gate is hardcoded on for this architecture, so make output_gate a required keyword argument end to end, drop the None specialization branches from both kernels, and assert gate contiguity. This leaves exactly one kernel specialization per profile”——这是本 PR 最有价值的一次讨论，把「内核参数列表变更必须同步 warmup 桩」这条隐性契约固定下来。
- coderabbitai 提出 3 条 Minor (均标 ④ Minor / ④ Quick win)：① 测试容差过松，rtol/atol=2e-2 无法区分「先乘 gate 再舍入」与「先舍入再乘 gate」，建议补 split 配置 (1, 24, 2, 1792) 与 direct-write 配置 (513, 6, 1, 1024) 的 BF16 边界用例；② ops/ple.py 中 mixed batch 的 graph padding 行不在 spec_token_indx/non_spec_token_indx 内，ple_conv 只启动 token_indices.numel() 个 program，padding 行可能停在 residual 而非 outer_residual + residual；③ output_gate_view = output_gate.view_as(q) 允许末维非单位 stride，建议拒绝或强制连续并断言 stride(2) == 1。
- 第 ③ 条被维护者的修复部分吸收 (assert output_gate.is_contiguous() 已隐含末维 stride 为 1，并删除了 None 分支)；第 ①② 条在提供的材料里没有作者回复或后续 commit，PR 直接合入 (ZJY0516 APPROVED)，属于遗留待验证项。
- claude[bot] 因 PR 来自 fork 自动评审被关闭，未产生实质意见；mergify[bot] 曾提示需 rebase 解决冲突，最终以 merge branch 'main' 收尾。
- 融合后 warmup 位置参数错位导致启动期 TypeError (correctness)：已修复：warmup 两次调用均补传 gate 张量及其 row/head stride，gate 变为必填参数，`assert output_gate.is_contiguous()` 生效。

- output_gate 视图的末维 stride 未被约束 (correctness): 部分采纳: 维护者用必填参数 + `assert output_gate.is_contiguous()` 消除了 None 分支并隐含末维 stride 为 1, 但没有显式加 `stride(2) == 1` 断言, 也没有改写成连续化回退。
- mixed batch 的 graph padding 行是否会漏加 outer_residual (correctness): 未决: 提供的材料中没有作者回复或后续 commit, PR 仍被合入; 需要在 spec + padding 混合批次上补逐位用例确认。
- BF16 舍入顺序缺少边界敏感的断言 (testing): 未决: 材料中未见新增用例或作者回应; 当前测试无法守住 PR 声明的舍入边界契约。

风险与影响

- 风险:
 1. 内核参数列表与 warmup 的隐性耦合 (已修但机制仍在): ops/qa.py 的 `warmup_qsa_sparse_paged_attention` 用位置参数镜像 kernel 签名, 本次融合后未同步, 导致 `enable_jit_warmup=True` (默认) 时启动即 `TypeError`; 若再有内核签名变更而漏改 warmup, 会重复同类启动级故障, 类型检查无法捕获。
 2. graph padding / mixed spec batch 的残差可能漏加 (未验证): coderabbitai 指出 padding 行不在任何 token index 映射中时不会执行 outer_residual 的 load; 而 tests/models/qwen4_exp/test_ple.py 的 graph_padding 断言却期望 (outer_residual + residual) 成立。两者结论冲突且没有作者回复, 若 coderabbitai 判断成立, spec decode + padding 组合下会少加一项 hidden_states, 属于静默数值偏差而非崩溃。
 3. 舍入顺序的测试强度不足: test_qsa_reference.py 用 `rtol=2e-2, atol=2e-2` 校验, 而 kernel 里 BF16 舍入与 `tl.sigmoid` 的先后顺序只带来约 $1e-3$ 量级差异, 现有用例无法守住 PR body 声明的 “preserve the original BF16/FP16 rounding boundaries”, 后续重构可能悄悄改变数值而不被测试发现。
 4. 布局假设依赖调用方守约: qsa_sparse_paged_attention 现在要求 output_gate 连续, 否则断言失败; ple_conv 要求 outer_residual 与 residual 行序、切片完全一致 (num_actual_tokens 裁剪), 一旦上层传入未按同一 token 顺序排列的张量, 会以错误行做加法。
 5. 性能收益随 batch 收敛: PLE 在 $\text{batch} \geq 256$ 时仅 1.02x、QSA 在 513 行仅 1.01x, 收益集中在低并发解码与小 prefill, 高吞吐场景的端到端提升有限, 不宜按 1.44x 外推。
 6. 影响面受限于单模型单后端: 改动全部位于 vllm/models/qwen4_exp/nvidia/, 不触及 KV cache 格式、权重加载或其他模型路径, 回归风险主要集中在 Qwen4Exp 自身。
 - 影响: 用户 / 线上服务: Qwen4Exp (Qwen3.8-Flash-Next) 在 NVIDIA 后端的单请求延迟改善明显——batch=1 解码 PLE 快 1.44x、QSA 门控快 1.105x, 对延迟敏感的首 token 与小 prefill 场景收益最大; 高并发吞吐场景提升收敛到 1%-2%。由于改动保持数值等价, 输出分布 (除浮点末位) 不变, 可安全替换。系统与工程: 无新增配置、无权重/KV cache 契约变更、无部署步骤变化, 融合后少两个逐点 kernel launch, CUDA graph 捕获的节点数减少。团队: 本 PR 确立了该模型族的两条工程惯例——融合点必须先复现原 eager 的舍入边界, 内核签名变更必须同步 warmup 桩与测试; 同时暴露了「Triton 位置参数漂移只能靠启动期 warmup 触发」的脆弱点。
 - 风险标记: 内核签名变

更易漏改 warmup, 曾致默认启动路径崩溃 (已修), 图填充行残差路径存疑, 舍入顺序测试偏弱, 布局假设依赖调用方守约

关联脉络

- PR #55272 (PR body 中提及, 未在提供的上下文中给出标题): PR body 明确的互补 PR: 它移除 model-level compile/custom-op plumbing, 但当前 diff 仍把 PLE 外层残差与 QSA gate 作为独立 PyTorch 运算, 本 PR 若在其后落地需 rebase 到其 breakable-graph 调用结构。
- PR #48498 [Performance] Add Triton kernel for Gemma3n sparse GELU: 同类「把逐点 / 激活运算并入 Triton 内核」的性能改造: 同样要求数值等价、配套内核级单测与基准脚本, 可作为融合范式与测试强度的对照。
- PR #55737 [Perf][GLM-5.3-Flash] Use FlashKDA for KDA chunked prefill (1.7-3.8x faster than the Triton chunk path): 同一仓库中逐模型内核特化性能工作的近期样本, 反映「按模型定制内核 + 保留数值边界 + 附 microbenchmark」的演进节奏。
- PR #56741 [Refactor] Normalize DeepSeek V4.1 model package naming: 与 vllm/models/qwen4_exp/nvidia/ 相同的「models/< 模型 >/< 厂商 >/」新包布局一脉相承, 说明 Qwen4Exp 采用了仓库正在推行的模型包组织方式。