PR #33455 完整报告

sgl-project/sglang

Revert "Add flashinfer rmsnorm + quant fusion support SM90, SM100, SM120"

合并时间：2026-08-04 10:03

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/33455

执行摘要

- 一句话：回滚 flashinfer rmsnorm+FP8 量化融合，修复 CI 失败
- 推荐动作：这是一次典型的“快速止血”型 revert，值得快速浏览以理解 kernel 特性合入在 CI 层面的依赖风险：新 kernel 合入前必须确保 CI 先构建并安装对应 sgl-kernel，再运行依赖它的测试。对量化路径优化感兴趣的同学，建议精读被回滚 diff 中 layernorm.py 的融合入口设计、_fp8_static_input_scale 的识别逻辑以及 apply_fp8_linear 的 pre_quant_output_dtype 契约——它们是将来的 #33469/#33471 重新合入时最有价值的设计资产。

功能与动机

PR body 直接声明原提交破坏 CI：Reverts sgl-project/sglang#32994，并给出失败 CI run 链接。原 PR 作者 DevashishLal-CB 在关联 issue 评论中称该 CI run 没有构建最新 sgl-kernel，命中的 assert 在最新 kernel 上不会触发、本地测试通过；但维护者仍选择整体 revert，以最快速度恢复 main 分支的绿色状态，让特性后续以拆分形式重新合入。

实现拆解

1. 提交级回滚：对 #32994 的 merge commit（39609837537fbae03eb148339d2406651bac8ba4）执行 git revert，生成单一回滚提交（head_sha 9b3d496），一次性还原 14 个文件的变更。
2. 归一化层回滚（python/sglang/srt/layers/layernorm.py）：删除 flashinfer_rmsnorm_quant_available 标志、rmsnorm_quant/fused_add_rmsnorm_quant 导入、_fp8_static_input_scale、_is_static_per_tensor_fp8_linear 辅助函数以及 forward_with_per_tensor_quant_fusion 方法；RMSNorm 各 forward* 签名移除 quant_linear 参数，forward_cuda 中置于 batch-invariant 守卫之后的融合路由块被整段删除，恢复纯归一化行为。
3. FP8 GEMM 路径回滚（fp8_utils.py、fp8.py、compressed_tensors_w8a8_fp8.py）：apply_fp8_linear 移除 pre_quant_output_dtype 参数与 input_prequantized 分支、删除 native scalar-a 判断（含 _is_sm90_supported），输出 dtype 统一恢复为 input.dtype；Fp8LinearMethod.apply 与 CompressedTensorsW8A8Fp8.apply_weights 删除对 (fp8, scale[, dtype]) 预量化 tuple 输入的接收逻辑，下游线性层不再消费预量化激活。
4. 模型调用契约恢复（llama.py、qwen2.py、llama_eagle.py、qwen2_eagle.py）：DecoderLayer.forward 中 input_layernorm / post_attention_layernorm 调用去掉

quant_linear=self.self_attn.qkv_proj 与 mlp.gate_up_proj 传参，恢复标准前向接口。

5. 配套与 AOT 回滚：删除融合特性专属的 benchmark（benchmark/kernels/bench_fused_rmsnorm_fp8_quant.py）与测试（test/registered/layers/test_layernorm_fusion.py）；test_fp8_utils.py 移除 scale 形状分派与预量化 dtype 用例，aot/tests/test_fp8_gemm.py 移除 native scalar-a 相关用例；aot/csrc/gemm/fp8_gemm_kernel.cu 回退 scalar-a CUTLASS kernel，aot benchmark 移除 scalar-a providers。本次没有新增 CI 配置或部署脚本，CI 恢复完全依赖代码回滚后重新触发全量测试。

关键文件：

- python/sglang/srt/layers/layernorm.py（模块 归一化层；类别 source；类型 dependency-wiring；符号 fp8_static_input_scale, _is_static_per_tensor_fp8_linear, forward_with_per_tensor_quant_fusion, RMSNorm.forward_cuda）：融合特性的核心入口所在：删除 flashinfer rmsnorm_quant 可用性标志、_fp8_static_input_scale / _is_static_per_tensor_fp8_linear 辅助函数、forward_with_per_tensor_quant_fusion 方法，并还原所有 forward* 的 quant_linear 参数与融合路由块，是本次回滚的主战场。
- python/sglang/srt/layers/quantization/fp8_utils.py（模块 FP8 工具；类别 source；类型 dependency-wiring；符号 apply_fp8_linear, pre_quant_output_dtype, _is_sm90_supported）：apply_fp8_linear 是融合激活消费端的核心：回滚删除 pre_quant_output_dtype 参数、input_prequantized 预量化分支与 native scalar-a 分派，out_dtype 恢复为 input.dtype，直接决定下游 GEMM 的调用契约。
- python/sglang/srt/layers/quantization/fp8.py（模块 量化方案；类别 source；类型 dependency-wiring；符号 Fp8LinearMethod.apply）：Fp8LinearMethod.apply 删除对预量化 (fp8, scale, dtype) tuple 输入的分支，恢复只接受普通 bf16/fp16 激活的旧接口。
- python/sglang/srt/layers/quantization/compressed_tensors/schemes/compressed_tensors_w8a8_fp8.py（模块 压缩量化；类别 source；类型 core-logic；符号 CompressedTensorsW8A8Fp8.apply_weights）：compressed-tensors W8A8-FP8 方案的 apply_weights 同样删除 tuple 预量化输入处理，与 fp8.py 保持一致的契约回退。
- python/sglang/srt/models/llama.py（模块 模型前向；类别 source；类型 data-contract；符号 LlamaDecoderLayer.forward）：DecoderLayer.forward 中 input_layernorm / post_attention_layernorm 调用移除 quant_linear 传参，是融合入口在真实模型路径上的接线点。
- python/sglang/srt/models/qwen2.py（模块 模型前向；类别 source；类型 data-contract；符号 Qwen2DecoderLayer.forward）：与 llama.py 相同，Qwen2 DecoderLayer 的归一化调用恢复为标准接口，避免融合参数残留。
- test/registered/layers/test_layernorm_fusion.py（模块 融合测试；类别 test；类型 test-coverage；符号 TestRMSNormFp8QuantFusion, test_rms_norm_fp8_quant_fusion, test_forward_cuda_quant_linear_dispatch）：融合特性专属测试被整体删除，该文件覆盖融合正确性与前向 dispatch 条件，是回滚后回归保护缺口的主要体现。
- test/registered/quant/test_fp8_utils.py（模块 FP8 测试；类别 test；类型 test-coverage；符号 TestApplyFp8LinearScaleDispatch, TestApplyFp8LinearPrequantOutputDtype）

：移除 apply_fp8_linear 的 scale 形状分派测试与预量化输出 dtype 传播测试，这两个用例专门验证新契约。

- python/sglang/kernels/aot/tests/test_fp8_gemm.py（模块 GEMM 测试；类别 test；类型 test-coverage；符号 _native_scalar_a_supported, test_scalar_a_channelwise_b, test_rejects_invalid_a_scale_count, test_rejects_scalar_a_with_multiple_rows_on_sm89）：移除 native scalar-a 支持探测与 fp8_scaled_mm 的 scalar-a 相关用例，与 AOT kernel 回退配套。
- benchmark/kernels/bench_fused_rmsnorm_fp8_quant.py（模块 基准脚本；类别 source；类型 deletion；符号 make_layer, run_unfused, run_fused_default, run_fused_cute）：融合特性专属微基准被整体删除，此前用于对比 unfused / flashinfer default / CuTe-DSL 三路性能。
- python/sglang/kernels/aot/csrc/gemm/fp8_gemm_kernel.cu（模块 AOT 内核；类别 other；类型 core-logic）：AOT CUTLASS kernel 回退 native scalar-A 支持，这是 #32994 中与 sgl-kernel 构建关联最紧密的部分，也是 CI 失败疑点所在。

关键符号：RMSNorm.forward_cuda, RMSNorm.forward_with_per_tensor_quant_fusion, _fp8_static_input_scale, _is_static_per_tensor_fp8_linear, apply_fp8_linear, Fp8LinearMethod.apply, CompressedTensorsW8A8Fp8.apply_weights, LlamaDecoderLayer.forward, Qwen2DecoderLayer.forward

关键源码片段

python/sglang/srt/layers/layernorm.py

融合特性的核心入口所在：删除 flashinfer rmsnorm_quant 可用性标志、fp8_static_input_scale / _is_static_per_tensor_fp8_linear 辅助函数、forward_with_per_tensor_quant_fusion 方法，并还原所有 forward* 的 quant_linear 参数与融合路由块，是本次回滚的主战场。

```
def forward_cuda(
    self,
    x: torch.Tensor,
    residual: Optional[torch.Tensor] = None,
    post_residual_addition: Optional[torch.Tensor] = None,
) -> Union[torch.Tensor, Tuple[torch.Tensor, torch.Tensor]]:
    # 回滚后：quant_linear 参数已被移除，RMSNorm 不再感知下游 FP8 线性层，
    # 融合路径（forward_with_per_tensor_quant_fusion）整段删除。
    if x.numel() == 0:
        # 空输入短路返回，避免触发 kernel；residual 语义保持不变。
        if residual is not None:
            if post_residual_addition is not None:
                residual = residual + post_residual_addition
            return x, residual
        return x

    # sgl_kernel 的 rmsnorm 只接受 2D 输入，高维张量先展平再恢复形状
    needs_reshape = x.dim() != 2 and residual is None
```

```
    if needs_reshape:
        original_shape = x.shape
        x = x.contiguous().reshape(-1, original_shape[-1])

    # 自定义方差维度与融合 kernel 不兼容，直接走 native 实现
    if self.variance_size_override is not None:
        return self.forward_native(x, residual, post_residual_addition)

    # batch-invariant 模式同样跳过；原融合路由位于此守卫之后，回滚后已不存在
    if is_batch_invariant_mode_enabled():
        if residual is not None or self.cast_x_before_out_mul:
            return self.forward_native(x, residual, post_residual_addition)
        out = rms_norm_batch_invariant(x, self.weight.data, self.variance_epsilon)
        if needs_reshape:
            out = out.reshape(original_shape)
        return out
    ...
```

python/sglang/srt/layers/quantization/fp8_utils.py

apply_fp8_linear 是融合激活消费端的核心：回滚删除 pre_quant_output_dtype 参数、input_prequantized 预量化分支与 native scalar-a 分派，out_dtype 恢复为 input.dtype，直接决定下游 GEMM 的调用契约。

```
def apply_fp8_linear(
    input: torch.Tensor,
    weight: torch.Tensor,
    weight_scale: torch.Tensor,
    input_scale: Optional[torch.Tensor] = None,
    input_scale_ub: Optional[torch.Tensor] = None,
    bias: Optional[torch.Tensor] = None,
    cutlass_fp8_supported: bool = cutlass_fp8_supported(),
    use_per_token_if_dynamic: bool = False,
    pad_output: Optional[bool] = None,
    compressed_tensor_quant: bool = False,
) -> torch.Tensor:
    # 回滚后：pre_quant_output_dtype 参数、input_prequantized 预量化分支、
    # native scalar-a 分派（_is_sm90_supported 等）全部移除，
    # 输出 dtype 恢复为 input.dtype。
    if pad_output is None:
        # torch._scaled_mm 在 batch>16 时对 padding 更友好，torch.compile 除外
        pad_output = not cutlass_fp8_supported and not get_bool_env_var(
            "SGLANG_ENABLE_TORCH_COMPILE"
        )
    output_padding = 17 if pad_output else None

    # 统一按 2D 处理，输出形状保持原始 rank
    input_2d = input.view(-1, input.shape[-1])
    output_shape = [*input.shape[:-1], weight.shape[1]]
```

```
if compressed_tensor_quant:
    # compressed-tensors 路径：按需 padding，静态 scale 时优先交给
    # inductor 与 RMSNorm/residual 融合，减少 kernel 启动次数
    num_token_padding = output_padding
    if cutlass_fp8_supported and weight_scale.numel() == weight.shape[1]:
        num_token_padding = None
    if (
        input_scale is not None
        and input_scale.numel() == 1
        and get_exec().graph.cuda_graph_config.prefill.tc_compiler == "inductor"
    ):
        qinput = (
            (input_2d * input_scale.reciprocal())
            .clamp(min=fp8_min, max=fp8_max)
            .to(fp8_dtype)
        )
        x_scale = input_scale
    else:
        qinput, x_scale = scaled_fp8_quant(
            input_2d,
            input_scale,
            num_token_padding=num_token_padding,
            use_per_token_if_dynamic=use_per_token_if_dynamic,
        )
else:
    # cutlass w8a8 fp8 sgl-kernel 只支持 per-token scale;
    # per-tensor scale 需要广播成逐行 scale 才能进入 fp8_scaled_mm
    if input_scale is not None:
        assert input_scale.numel() == 1
        qinput, x_scale = static_quant_fp8(
            input_2d, input_scale, repeat_scale=cutlass_fp8_supported
        )
    else:
        # 默认走动态 per-token 量化
        if _is_cuda:
            qinput, x_scale = sglang_per_token_quant_fp8(input_2d)
        elif _is_hip and weight_scale.numel() == 1:
            qinput, x_scale = scaled_fp8_quant(
                input_2d,
                input_scale,
                use_per_token_if_dynamic=use_per_token_if_dynamic,
            )
        else:
            qinput, x_scale = per_token_group_quant_fp8(
                input_2d, group_size=input_2d.shape[1]
            )
...
```

评论区精华

PR 本身没有 review 评论，核心讨论发生在关联 issue #32994 的评论中：原 PR 作者 DevashishLal-CB 指出 CI run 30868463119 没有构建最新 sgl-kernel，命中的 assert 在最新 kernel 上已经不会触发（他修改过该断言），并在本地通过了 pytest ./test/registered/quant/test_modelopt_fp8.py。分歧点在于：问题根源是 CI 构建顺序 / 缓存问题，而非融合特性本身的正确性。维护者的决策是整体 revert（本 PR），先恢复 main 绿色，再让特性按原计划拆分（#33469/#33471）重新合入。

- CI 失败根因：sgl-kernel 构建顺序 vs 特性本身 (question): 维护者选择整体 revert 以恢复 main 分支稳定，特性后续需以拆分形式（原计划 #33469/#33471）重新合入。

风险与影响

- 风险：
  1. 性能回退：根据 #32994 的 benchmark，回滚后 H100 约 1.5%~2.5%、B200 约 4%~5% 的端到端收益暂时丢失，FP8 静态 per-tensor 量化模型恢复为 RMSNorm 与量化两次 kernel 执行。
  2. API 契约回归：layernorm.py 的 forward_* 签名、apply_fp8_linear 签名、Fp8LinearMethod.apply 与 CompressedTensorsW8A8Fp8.apply_weights 的输入约定全部恢复旧版；若 revert 合入前已有未同步的外部分支依赖新接口，会出现不一致，仓库内调用点本次已一并还原。
  3. 测试覆盖移除：test_layernorm_fusion.py 整体删除，test_fp8_utils.py 与 test_fp8_gemm.py 中融合与 scalar-a 用例移除，融合路径暂时没有回归保护，后续重新合入时必须补齐。
  4. CI 依赖未根治：失败根因疑似 CI 未构建最新 sgl-kernel，本 PR 只做了止血，未修改构建顺序；同类 kernel 特性后续合入时仍可能复现同类问题。
  5. 重新合入冲突风险：如果 main 分支后续出现依赖新接口的改动，重新合入 #32994 的内容会产生冲突，需要额外协调。 - 影响：影响范围集中在 FP8 量化模型的前向路径：python/sglang/srt/layers/layernorm.py、quantization/fp8_utils.py、quantization/fp8.py、compressed_tensors_w8a8_fp8.py 以及 llama.py/qwen2.py 及其 eagle 变体的 DecoderLayer 调用契约均回退到融合前状态。对用户而言，使用原生 FP8 静态 per-tensor 量化或 compressed-tensors W8A8-FP8 的模型会失去量化融合带来的吞吐收益；对团队而言，需要重新规划 #32994 的合入流程，优先解决 CI 对 sgl-kernel 构建顺序的依赖，并按 AOT kernel 与融合逻辑两部分拆分落地。整体影响是暂时性的性能回退换取主线稳定，影响程度中等。 - 风险标记：核心量化路径回滚，性能收益暂时丢失，特性重新合入有冲突风险，CI 依赖 sgl-kernel 构建顺序，融合路径测试覆盖移除

关联脉络

- PR #32994 Add flashinfer rmsnorm + quant fusion support SM90, SM100, SM120: 本 PR 正是对该 PR 的 revert；其声称 H100 1.5%~2.5%、B200 4%~5% 的性能收益，且因 CI 失败被回滚。