PR #33471 完整报告

sgl-project/sglang

runtime: Add flashinfer rmsnorm + quant fusion support SM90, SM100, SM120- #32994

合并时间：2026-08-09 20:15

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/33471

执行摘要

- 一句话：新增 flashinfer RMSNorm+FP8 量化融合，支持 SM90/100/120
- 推荐动作：值得精读 PR，尤其关注 layernorm.py 中如何用特征探测安全地启用融合路径、fp8_utils.py 如何通过 pre_quant_output_dtype 保持 dtype 契约。该模式可作为后续 kernel fusion 接入的参考模板。注意目前仅 LLaMA / Qwen2 两个模型接入，若有相关模型需求可参照推进。

功能与动机

PR body 说明这是 #32994 的 runtime changes，即上游 flashinfer 新增了 rmsnorm_quant 融合内核，但 SGLang 运行时尚未接入。通过接入融合路径，可避免 RMSNorm 输出先写回显存、再由 static_quant_fp8 重新读取并量化的两步开销，降低 kernel launch 次数和显存带宽占用。

实现拆解

实现拆解

1. 探测 flashinfer 融合内核：在 python/sglang/srt/layers/layernorm.py 的模块加载阶段新增 _flashinfer_rmsnorm_quant_available 标志，尝试导入 flashinfer.norm.rmsnorm_quant 与 fused_add_rmsnorm_quant，失败则置 False 并继续走原有非融合路径。
2. 新增融合可行性判定：新增 _fp8_static_input_scale(linear) 与 _is_static_per_tensor_fp8_linear(quant_method, linear) 两个辅助函数，识别原生 Fp8LinearMethod（排除 block/mxfp8/marlin）以及 compressed-tensors W8A8-FP8 静态 per-tensor 输入方案，并确认 input_scale 为单元素张量后返回该 scale。
3. 扩展 RMSNorm 前向接口：为 forward_cuda 等所有后端 forward 方法增加可选参数 quant_linear，并在 forward_cuda 的常规路径中（排除空输入、variance override、batch-invariant、HF cast 等不兼容分支）调用新增的 forward_with_per_tensor_quant_fusion 方法。该方法基于 flashinfer 融合内核计算 (fp8, scale, orig_dtype) 或 ((fp8, scale, orig_dtype), residual_out)，其中 orig_dtype 用于下游 GEMM 正确输出模型原始 dtype。
4. 打通 FP8 linear 预量化输入：在 apply_fp8_linear 中新增 pre_quant_output_dtype 参数，检测到输入已是 FP8 时跳过再量化、复用传入的 per-tensor scale，并按该参数或 bf16

决定输出 dtype；在 Fp8LinearMethod.apply 与 CompressedTensorsW8A8Fp8.apply_weights 中增加对 tuple 输入 (fp8_input, input_scale[, orig_dtype]) 的分派。

5. 模型入口接线：在 llama.py 与 qwen2.py 的 forward 中将 self.self_attn.qkv_proj / self.mlp.gate_up_proj 作为 quant_linear 传入两层 layernorm；llama_eagle.py 与 qwen2_eagle.py 中的 stub lambda 同步增加 quant_linear 参数，避免调用签名不一致。
6. 测试与基准：新增 test/registered/layers/test_layernorm_fusion.py 覆盖数值正确性、输出契约及 forward_cuda 分派条件；扩展 test/registered/quant/test_fp8_utils.py 覆盖不同 SM capability 下的 scale 形状与预量化输入路径；新增 benchmark/kernels/bench_fused_rmsnorm_fp8_quant.py 对比 unfused / fused / fused_cute 三种实现。

关键文件：

- python/sglang/srt/layers/layernorm.py（模块 归一化层；类别 source；类型 core-logic；符号 _fp8_static_input_scale, _is_static_per_tensor_fp8_linear, forward_with_per_tensor_quant_fusion）：核心变更文件：新增 flashinfer 融合内核探测、静态 per-tensor FP8 判定、forward_with_per_tensor_quant_fusion 融合前向方法，并扩展所有后端 forward 接口。
- python/sglang/srt/layers/quantization/fp8.py（模块 FP8 量化；类别 source；类型 dependency-wiring；符号 Fp8LinearMethod.apply）：修改 Fp8LinearMethod.apply 支持预量化元组输入，并调整相关 import 格式。
- python/sglang/srt/layers/quantization/fp8_utils.py（模块 FP8 工具；类别 source；类型 core-logic；符号 apply_fp8_linear）：apply_fp8_linear 新增预量化输入处理与 pre_quant_output_dtype 参数，是 dtype 契约的关键落点。
- python/sglang/srt/layers/quantization/compressed_tensors/schemes/compressed_tensors_w8a8_fp8.py（模块 压缩张量；类别 source；类型 core-logic；符号 CompressedTensorsW8A8Fp8.apply_weights）：compressed-tensors W8A8 FP8 方案同样支持预量化元组输入，保持与原生 FP8 路径一致。
- python/sglang/srt/models/llama.py（模块 LLaMA 模型；类别 source；类型 data-contract；符号 LlamaForCausalLM.forward）：将 qkv_proj / gate_up_proj 作为 quant_linear 传入 layernorm，是模型侧接入融合路径的示例。
- python/sglang/srt/models/qwen2.py（模块 Qwen2 模型；类别 source；类型 data-contract；符号 Qwen2ForCausalLM.forward）：同 llama.py，为 Qwen2 系列接入融合路径。
- test/registered/layers/test_layernorm_fusion.py（模块 融合测试；类别 test；类型 test-coverage；符号 TestRMSNormFp8QuantFusion）：新增融合路径单元测试，覆盖数值正确性、输出契约与 forward_cuda 分派条件。
- benchmark/kernels/bench_fused_rmsnorm_fp8_quant.py（模块 内核基准；类别 test；类型 test-coverage；符号 make_layer, make_inputs, run_unfused, _run_fused）：新增融合 vs 非融合 vs CuTe-DSL 的微基准，用于验证提速效果并辅助选择内核。

关键符号：_fp8_static_input_scale, _is_static_per_tensor_fp8_linear, forward_with_per_tensor_quant_fusion, apply_fp8_linear, Fp8LinearMethod.apply, CompressedTensorsW8A8Fp8.apply_weights, LlamaForCausalLM.forward, Qwen2ForCausalLM.forward

关键源码片段

python/sglang/srt/layers/layernorm.py

核心变更文件：新增 flashinfer 融合内核探测、静态 per-tensor FP8 判定、forward_with_per_tensor_quant_fusion 融合前向方法，并扩展所有后端 forward 接口。

```
def _fp8_static_input_scale(linear) -> Optional[torch.Tensor]:
    """返回可消费预量化输入的静态 per-tensor FP8 线性层输入 scale，否则返回 None。"""
    if linear is None:
        return None
    quant_method = getattr(linear, "quant_method", None)
    if quant_method is None:
        return None
    # 仅接受原生 Fp8LinearMethod（非 block/mxfp8/marlin）或 compressed-tensors W8A8-FP8 静态输入方案
    if not _is_static_per_tensor_fp8_linear(quant_method, linear):
        return None
    input_scale = getattr(linear, "input_scale", None)
    # flashinfer 融合内核只支持 per-tensor 量化，scale 必须为单元素
    if input_scale is None or input_scale.numel() != 1:
        return None
    return input_scale


def _is_static_per_tensor_fp8_linear(quant_method, linear) -> bool:
    """判断量化方法是否为静态 per-tensor FP8 线性层。"""
    try:
        from sglang.srt.layers.quantization.fp8 import Fp8LinearMethod
    except ImportError:
        Fp8LinearMethod = ()
    if isinstance(quant_method, Fp8LinearMethod):
        # 排除 block / MXFP8 / Marlin 等非 per-tensor 方案
        return not (
            getattr(quant_method, "block_quant", False)
            or getattr(quant_method, "use_mxfp8", False)
            or getattr(quant_method, "use_marlin", False)
        )
    try:
        from sglang.srt.layers.quantization.compressed_tensors.compressed_tensors import (
            CompressedTensorsLinearMethod,
        )
        from sglang.srt.layers.quantization.compressed_tensors.schemes import (
            CompressedTensorsW8A8Fp8,
```

```
        )
    except ImportError:
        return False
    if isinstance(quant_method, CompressedTensorsLinearMethod):
        scheme = getattr(linear, "scheme", None)
        return isinstance(scheme, CompressedTensorsW8A8Fp8) and getattr(
            scheme, "is_static_input_scheme", False
        )
    return False
```

python/sglang/srt/layers/quantization/fp8.py

修改 Fp8LinearMethod.apply 支持预量化元组输入，并调整相关 import 格式。

```
def apply(self, layer, x, bias=None):
    # ... 此前 block_quant 分支 ...
    if isinstance(x, tuple):
        # 来自 fused RMSNorm + FP8 quant 内核的预量化激活：
        # x = (fp8_input, per_tensor_input_scale[, orig_dtype])
        # apply_fp8_linear 会识别 FP8 dtype 并跳过再量化
        qx, x_scale = x[0], x[1]
        out_dtype = x[2] if len(x) > 2 else None
        return apply_fp8_linear(
            input=qx,
            weight=layer.weight,
            weight_scale=layer.weight_scale,
            input_scale=x_scale,
            bias=bias,
            cutlass_fp8_supported=self.cutlass_fp8_supported,
            use_per_token_if_dynamic=self.use_per_token_if_dynamic,
            pre_quant_output_dtype=out_dtype,
        )
    return apply_fp8_linear(
        input=x,
        weight=layer.weight,
        weight_scale=layer.weight_scale,
        input_scale=layer.input_scale,
        bias=bias,
        cutlass_fp8_supported=self.cutlass_fp8_supported,
        use_per_token_if_dynamic=self.use_per_token_if_dynamic,
    )
```

评论区精华

PR 无实质性 review 评论，仅由 BBuf 批准并标注 LGTM。由于是承接上游内核的运行时适配，讨论主要集中在 CI 状态，issue 评论中仅有 /tag-and-rerun-ci extra 与两次失败重跑记录，未发现关于设计取舍的讨论。

- 暂无高价值评论线程

风险与影响

- 风险：
  - 兼容性 / 依赖风险：融合路径强依赖 flashinfer 是否提供 rmsnorm_quant，检测失败会自动回退到非融合路径，不存在直接崩溃风险，但低版本 flashinfer 会静默失去优化。
  - dtype 契约风险：apply_fp8_linear 对预量化输入默认输出 bf16，若未正确传入 orig_dtype，FP16 模型可能出现输出 dtype 不匹配（如 attention 中 query/key dtype 不一致）。已有回归测试 TestApplyFp8LinearPrequantOutputDtype 覆盖。
  - 触发条件风险：forward_cuda 中仅当 quant_linear 非 None、未启用 HF cast 语义且 flashinfer 可用时才走融合路径；若未来新增模型漏传 quant_linear，则不会融合但行为仍正确。
  - 回归影响面：RMSNorm 所有后端 forward 方法均增加可选参数，接口向后兼容；但 llama.py / qwen2.py 的调用方式发生改变，需保证其余模型（如 qwen3、deepseek 等）暂时不受影响，它们仍走旧路径。
  - 影响：用户 / 系统：在支持的 NVIDIA GPU（SM90/SM100/SM120）上，配合新版本 flashinfer，LLaMA / Qwen2 系列的 FP8 静态 per-tensor 量化推理可减少一次激活量化 kernel 与一次中间显存往返，预期降低 prefill/decode 延迟；其余场景行为不变。开发者：引入 (fp8, scale, orig_dtype) 元组约定与 quant_linear 参数，后续新增模型可参照 llama.py 的接线方式接入融合路径，但需同步保障所有 forward 后端签名一致。测试：新增的单元测试全部在 base-b 阶段（1-gpu-large）运行，CI 时长有所增加。
- 风险标记：依赖 flashinfer 新内核，部分模型接入，新增 API 契约

关联脉络

- 暂无明显关联 PR