PR #32843 完整报告

sgl-project/sglang

[Quant] Keep the flashinfer_deepgemm FP8 GEMM to 1 <= M < 32

合并时间：2026-08-02 00:36

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/32843

执行摘要

- 一句话：flashinfer_deepgemm FP8 GEMM 限定在 1 <= M < 32
- 推荐动作：值得精读。这是一个小改动但论证极其扎实的 PR：用微基准、准确率实验、路由矩阵和失败模式分析支撑一个 20 行改动，并诚实地划清“性能论据为主、准确率论据为辅”的边界。值得关注的设计决策包括：单一 triton 出口折叠 M 检查、DeepGEMM 自行兜底、以及因为 DeepEP 交互问题主动回退 auto 默认。对做 kernel 调度和后端选择的工程师尤其有参考价值。

功能与动机

PR body 明确指出性能是主要理由，且无条件成立：H200 微基准显示 M < 32 时 flashinfer 比 DeepGEMM 快 1.05x，M 在 32-64 时慢 1.42x（最差区域），M=850 时慢 1.10x；因此该回退不是权衡，而是全区间占优的分裂。精度是次要理由，Qwen3.6-27B-FP8 上 GSM8K 掉 3.8 分，但作者强调这与 checkpoint 相关，现有精度测试在 Qwen3-4B 上测不出差异。此外 M == 0 在 fp8_blockscale_gemm_sm90 中会直接崩溃（input_ptr != nullptr 检查失败），而 DP attention 下空闲 rank 的零 token forward（ScheduleBatch.prepare_for_idle）是正常稳态输入，必须避免。

实现拆解

1. 在 flashinfer_deepgemm_w8a8_block_fp8_linear_with_fallback 中新增 M 守卫（文件 python/sglang/srt/layers/quantization/fp8_utils.py）：计算 m_supported = 1 <= input.view(-1, input.shape[-1]).shape[0] < 32，并在既有 shape/dtype 守卫前判断。M >= 32 且 deep_gemm_wrapper.ENABLE_JIT_DEEPGEMM 时直接调用 deepgemm_w8a8_block_fp8_linear_with_fallback，由 DeepGEMM 自行对不可服务形状做 triton 回退。
2. 折叠 M 检查进 triton 回退：将 m_supported 并入原有 shape_supported and dtype_supported 条件，使 M == 0 与 M >= 32（DeepGEMM 不可用时）走同一个 triton 出口，确保 UE8M0 scale 解包（_unpack_ue8m0_scale_for_triton）不丢失。这修复了早期版本在 DeepGEMM 关闭时 M >= 32 仍落入 flashinfer、以及 M == 0 时 triton 路径跳过 scale 解包的两个漏洞。
3. 回退 auto 模式默认切换：PR 演进过程中曾加入“Hopper 上 auto 模式默认启用 hybrid”的提交，后因 DeepEP 路径在 SGLANG_ENABLE_ASYNC_ASSERT=true 下触发 NaN 断言问题而被显式 revert（commit a4b3796）。最终 flashinfer_deepgemm 维持 opt-in 状态。

4. 测试与验证：PR 未新增测试文件，但依赖既有 test/registered/quant/test_fp8_blockwise_gemm.py 和 test/registered/ep/test_deepep_small.py 在 H200 上手工验证；路由矩阵 M in {0,1,8,31,32,64} 覆盖 DeepGEMM 开 / 关两种情形。所有测试均在 H200 上运行，CI 中唯一选择该后端的测试类受 skipIf(get_device_sm() != 90) 限制在 b200 runner 上是空操作，因此该路径需要人工确认。

关键文件：

- python/sglang/srt/layers/quantization/fp8_utils.py（模块 量化层；类别 source；类型 core-logic；符号 flashinfer_deepgemm_w8a8_block_fp8_linear_with_fallback）：核心改动文件：为 flashinfer_deepgemm 后端新增 1 <= M < 32 守卫，并修正 triton 回退的 scale 解包逻辑。

关键符号：flashinfer_deepgemm_w8a8_block_fp8_linear_with_fallback

关键源码片段

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

核心改动文件：为 flashinfer_deepgemm 后端新增 1 <= M < 32 守卫，并修正 triton 回退的 scale 解包逻辑。

```
# python/sglang/srt/layers/quantization/fp8_utils.py
# flashinfer_deepgemm_w8a8_block_fp8_linear_with_fallback 核心路由逻辑

def flashinfer_deepgemm_w8a8_block_fp8_linear_with_fallback(
    input: torch.Tensor,
    weight: torch.Tensor,
    block_size: List[int],
    weight_scale: torch.Tensor,
    input_scale: Optional[torch.Tensor] = None,
    bias: Optional[torch.Tensor] = None,
) -> torch.Tensor:
    assert input_scale is None

    output_dtype = input.dtype
    dtype_supported = output_dtype == torch.bfloat16

    # fp8_blockscale_gemm_sm90 要求 N % 64 == 0 且 K % 128 == 0
    shape_supported = weight.shape[0] % 64 == 0 and weight.shape[1] % 128 == 0

    # 关键守卫：flashinfer 的 fp8_blockscale_gemm_sm90 是单入口双内核，
    # 只有 M < 32 的 swapAB 内核值得使用；M >= 32 慢于 DeepGEMM，
    # M == 0 会因空指针检查在 kernel 内崩溃（DP attention 空闲 rank
    # 的零 token forward 是常态输入，见 ScheduleBatch.prepare_for_idle）。
    m_supported = 1 <= input.view(-1, input.shape[-1]).shape[0] < 32

    if not m_supported and deep_gemm_wrapper.ENABLE_JIT_DEEPGEMM:
        # DeepGEMM 覆盖 M == 0 和 M >= 32，且对无法服务的 shape
        # 会自行回退 triton，因此直接转发即可。
```

```
        return deepgemm_w8a8_block_fp8_linear_with_fallback(
            input, weight, block_size, weight_scale, input_scale, bias
        )

    if not (shape_supported and dtype_supported and m_supported):
        # 统一 triton 出口：保证 UE8M0 的 int32 scale 在这里被解包，
        # 避免 M == 0 或 shape 不合法时遗留未解包的 scale。
        if weight_scale.dtype == torch.int32:
            weight_scale = _unpack_ue8m0_scale_for_triton(
                weight_scale, weight.shape, block_size
            )
        return triton_w8a8_block_fp8_linear(
            input, weight, block_size, weight_scale, input_scale, bias
        )

    # 1 <= M < 32 且 shape/dtype 都满足时才真正走到 flashinfer 内核
    input_2d = input.view(-1, input.shape[-1])
    output_shape = [*input.shape[:-1], weight.shape[0]]

    output = fp8_blockscale_gemm_sm90(
        input_2d,
        weight,
        input_scale=None, # BF16 输入由内核内部量化
        weight_scale=weight_scale,
        out_dtype=output_dtype,
    )

    if bias is not None:
        output += bias
    return output.view(*output_shape)
```

评论区精华

1. DeepEP 交互问题与是否默认开启的分歧：mmangkad 质疑自动 dispatch 是否可保留，建议仅 DPA 时回退 DeepGEMM；zhendonghua 解释 DPA 下 flashinfer GEMM 会在 padding 行产生 NaN，CI 会检查 NaN，并已删除自动 dispatch；b8zhong 认为端到端性能收益不显著且会引入更多编译，倾向保持默认关闭。最终结论：保持 opt-in。
2. CI 失败与 run-ci 标签：test_deepep_small.py 在 4-gpu-h100 上多次失败，mmangkad 建议移除 run-ci 标签直到修复，最终通过 /rerun-test 在后续运行中通过。
3. 上流 bug 的定位：zhendonghua 将 DeepEP 交互中 NaN 的根因总结并单独立 issue（#33106），PR 内明确不做修复，并给出实证表格（不同后端对 uninitialized 行的不同反应窗口）。
4. 准确性缺陷的诚实验证：作者明确说明精度回退不通用，现有测试在 Qwen3-4B 上测不出差异，并给出二项标准误分析，声明性能论证才是支撑。

- 是否在 auto 模式默认启用 hybrid 后端 (design): 回退 auto 默认切换，flashinfer_deepgemm 保持 opt-in；DeepEP 交互问题单独立 issue 跟踪。

- CI 中 test_deepep_small.py 反复失败与 run-ci 标签 (testing): 通过 rerun 解决，run-ci 保留；但作者承认 CI 不覆盖 flashinfer_deepgemm 后端，守卫效果依赖人工 H200 验证。
- flashinfer_deepgemm 对 padding 行产生 NaN 的机制 (correctness): PR 不修复上流问题，作者在 #33106 单独立项；用户需避免 flashinfer_deepgemm 与 deepep 组合使用。

风险与影响

- 风险：
  1. CI 覆盖为空：该后端路径只在 SM90（H100/H200）上可达，唯一相关测试 TestFP8BlockwiseGemmFlashinferDeepGemm 注册在 4-gpu-b200 runner 上却因 skipIf(get_device_sm() != 90) 成为空操作，回归风险完全依赖人工验证。
  2. DP attention 下的 NaN 隐患未根除：DeepEP + flashinfer_deepgemm + 异步 NaN 断言组合会导致 server 启动失败，PR 明确不修复，仅提示用户不要组合使用；守卫无法保护 M < 32 时未初始化行进入内核的危险。
  3. 行为变更影响面：flashinfer_deepgemm 用户若依赖 M >= 32 时的 flashinfer 路径，切换后行为会变（转向 DeepGEMM），但该路径本身精度较差，且 PR 保留 triton 兜底。
  4. M 边界选择的普适性：守卫固定 M < 32，若未来 flashinfer 内核行为变化（如新增其他内核或阈值漂移），硬编码边界可能过时，需要同步更新。 - 影响：影响范围集中在量化后端的 GEMM 路由：使用 --fp8-gemm-backend flashinfer_deepgemm 的 Hopper 用户（FP8 块缩放模型）会观察到 M >= 32 时自动走 DeepGEMM，M == 0 时不再崩溃；M < 32 时保持 flashinfer 的 swapAB 加速。对默认配置（DeepGEMM auto）无影响，对社区用户而言修复了空 batch 崩溃并改善了 M >= 32 的精度 / 性能。团队影响：为后续类似后端守卫提供范本，暴露了 DeepEP 路径的上游内存初始化问题，并促使维护者意识到 CI 对 SM90 后端的覆盖缺口。 - 风险标记：测试覆盖缺口，上流未修复问题，硬编码阈值，后端行为变更

关联脉络

- PR #33106 issue: NaN with flashinfer_deepgemm + deepep (author-filed): PR 中发现的 DeepEP 交互 NaN 问题的上流跟踪 issue，PR 明确不修复并单独归档。
- PR #33128 Support DeepGEMM for standard MoE dispatch: 同为量化 /DeepGEMM 相关后端增强，扩展 DeepGEMM 在 MoE dispatch 路径中的使用，与本 PR 的 GEMM 路由改进互补。
- PR #32910 [DeepSeek-V4] Fix nvcc 13 crash building the topk_v2 kernel: 同为 JIT/ 量化内核修复，体现仓库近期对 DeepGEMM/JIT 路径稳定性的关注。