PR #35120 完整报告

sgl-project/sglang

[FlashInfer v0.6.18] add FlashInfer CuTe DSL NVFP4 W4A16 mode

合并时间：2026-09-01 09:47

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/35120

执行摘要

- 一句话：FlashInfer CuTe DSL 新增 NVFP4 W4A16 模式，激活保持 BF16
- 推荐动作：值得精读，尤其关注以下设计决策：quant_mode 作为贯穿 dense/MoE/dispatcher/ 模型守卫的统一决议值如何避免分叉；copy_or_rebind_param 在 CUDA graph 与 disk reload 场景下保持 tensor 绑定的手法；W4A16 下对 FlashInfer A2A 强制 BF16 dispatch 而非引入第二套 workspace 的取舍。本 PR 的测试组织（online 精度 + reload 确定性 + lm head 守卫）也值得作为量化后端集成的参考样例。

功能与动机

PR body 明确提出动机：

实现拆解

1. 新增环境开关与量化模式决议：在 python/sglang/srt/environ.py 中新增 SGLANG_FLASHINFER_CUTEDSL_NVFP4_W4A16（默认 false）；ModelOptFp4LinearMethod.__init__（modelopt_quant.py）依据该开关与 get_fp4_gemm_runner_backend().is_flashinfer_cutedsl() 计算 self.quant_mode（w4a16 或 w4a4）。dense 与 MoE 后续所有分支都由这个模式决议派生，保证两条路径配置一致。
2. Dense 路径改造：在 modelopt_quant.py 中，fp4_gemm 及 fake 实现增加 quant_mode 参数，input_sf 改为 Optional；w4a16 分支调用 FlashInfer 的 mm_bf16_fp4，不再需要激活 scale。process_weights_after_loading 在 W4A16 下改用 prepare_bf16_fp4_weights 预打包权重与 MMA-layout block scale，并用 copy_or_rebind_param 保持 weight/weight_scale_interleaved/alpha 的 Parameter 绑定，保证 disk reload 与 decode CUDA graph 的地址稳定性。apply 中 W4A16 分支保留 NVFP4_AWQ 的 pre_quant_scale 预缩放，随后直接以 BF16 输入调用 fp4_gemm。该改动覆盖 serialized ModelOpt 的 dense 线性层与 QKV 等。
3. MoE 路径改造：flashinfer_cutedsl.py 中，ensure_cutedsl_wrapper 将 quant_mode 传入 CuteDslMoEWrapper；CuteDslFp4MoeQuantInfo 增加 quant_mode 字段；fused_experts_none_to_flashinfer_cutedsl_fp4 与 fused_experts_flashinfer_to_flashinfer_cutedsl_fp4 在 W4A16 下让 BF16 激活直通（x_fp4 = hidden_states、x_sf = None、fc2_input_scale = None），跳过 fp4_quantize/nvfp4_quantize 与 FP4 布局 reshape。

refresh_cutedsl_standard_scales_for_weight_update 在 W4A16 下不再生成 per-token 全局 scale，GEMM alpha 只含权重反量化因子。

4. A2A/dispatcher 协调与配置校验：moe_hook.py 在 handle_a2a_moe 中新增 use_cutedsl_w4a16 判断：W4A16 且 FlashInfer A2A 时，显式设置 SGLANG_MOE_NVFP4_DISPATCH=1 直接抛 ValueError，且不再自动把 dispatch 置 true（保留 BF16 dispatch）；modelopt_quant.py 中 use_dispatch_fp4 追加 not use_cutedsl_w4a16 条件；CuTe DSL v1 DeepEP masked 路径在 W4A16 下显式 raise。最终语义为：W4A16 仅支持无 A2A 或 FlashInfer A2A（BF16 dispatch），DeepEP masked 路径保持 W4A4-only。
5. 模型级守卫与 LM head 识别：logits_processor.py 的 should_apply_lm_head_quant_method 增加 W4A16 runtime 状态识别（weight_scale_interleaved/alpha 等属性）；deepseek_v2.py 与 qwen3_5.py 的 GEMM+SwiGLU / silu+FP4 量化融合守卫均追加 quant_mode == "w4a4" 条件，防止 W4A16 误走 W4A4-only 融合 kernel。
6. 测试、文档与 CI 配套：新增 TestFlashinferCuteDSLMoeBackendNvFp4OnlineW4A16（nightly 4-gpu-b200，Nemotron-3-Super FP8 + 在线 NVFP4 量化，覆盖 FlashInfer A2A 与 fused finalize）、TestServerUpdateWeightsFromDiskNVFP4W4A16CuteDSL（TP4/DP4/EP4 无 A2A，两次 disk reload 后校验 decode 文本与 token logprobs 确定性）、test_lm_head_guard_accepts_modelopt_fp4_cutedsl_w4a16_runtime_state；test_flashinfer_cutedsl_dispatch.py 的 SimpleNamespace fixture 补声明 quant_mode="w4a4"（修复 main 合并后的 CI 漂移）；docs/docs/references/environment_variables.mdx 补充新环境变量说明。

关键文件：

- python/sglang/srt/layers/quantization/modelopt_quant.py（模块 量化层；类别 source；类型 data-contract；符号 fp4_gemm, ModelOptFp4LinearMethod, process_weights_after_loading, apply）：dense 路径核心改造：fp4_gemm 增加 quant_mode 分派、ModelOptFp4LinearMethod 支持 W4A16 权重预打包与 apply 分支，并影响 dispatch 决策。
- python/sglang/srt/layers/moe/moe_runner/flashinfer_cutedsl.py（模块 MoE 内核；类别 source；类型 core-logic；符号 ensure_cutedsl_wrapper, CuteDslFp4MoeQuantInfo, fused_experts_none_to_flashinfer_cutedsl_fp4, fused_experts_flashinfer_to_flashinfer_cutedsl_fp4）：MoE 路径核心改造：CuteDslMoEWrapper 传入 quant_mode，fused 函数在 W4A16 下直通 BF16 激活并跳过 FP4 布局处理。
- python/sglang/srt/arg_groups/moe_hook.py（模块 参数校验；类别 source；类型 core-logic；符号 handle_a2a_moe）：A2A 配置校验：W4A16 与 FlashInfer A2A 组合要求 BF16 dispatch，显式 NVFP4 dispatch 直接拒绝，防止静默走错量化路径。
- python/sglang/srt/layers/logits_processor.py（模块 输出头；类别 source；类型 core-logic；符号 should_apply_lm_head_quant_method）：LM head 量化守卫新增 W4A16 runtime 状态识别，防止 dense 输出头误用 W4A4 量化方法。

- python/sglang/srt/environ.py（模块 环境变量；类别 source；类型 configuration；符号 SGLANG_FLASHINFER_CUTEDSL_NVFP4_W4A16）：新增 SGLANG_FLASHINFER_CUTEDSL_NVFP4_W4A16 默认关闭的环境开关，是模式决议的源头。
- python/sglang/srt/models/deepseek_v2.py（模块 模型定义；类别 source；类型 data-contract）：DeepSeek 的 GEMM+SwiGLU 融合路径仅限 W4A4，避免 W4A16 下误走融合 kernel。
- python/sglang/srt/models/qwen3_5.py（模块 模型定义；类别 source；类型 data-contract；符号 _maybe_enable_silu_fp4_quant_fusion）：与 deepseek_v2.py 同类的 silu+FP4 量化融合守卫，review 中 mmangkad 指出后补上。
- test/registered/backends/test_flashinfer_nvfp4_online_moe_backend.py（模块 回归测试；类别 test；类型 test-coverage；符号 TestFlashinferCuteDSLMoeBackendNvFp4OnlineW4A16）：新增 nightly 4-gpu-b200 端到端 W4A16 在线量化精度测试，覆盖 FlashInfer A2A、DP attention 与 fused finalize。
- test/registered/rl/test_update_weights_from_disk_blackwell.py（模块 重载测试；类别 test；类型 test-coverage；符号 TestServerUpdateWeightsFromDiskNVFP4W4A16CuteDSL）：新增 W4A16 磁盘权重重载确定性测试，覆盖两次 reload + CUDA graph 释放 / 恢复后的 token 级一致性。
- test/registered/unit/model_loader/test_modelopt_loader.py（模块 单元测试；类别 test；类型 test-coverage；符号 test_lm_head_guard_accepts_modelopt_fp4_cutedsl_w4a16_runtime_state）：补充 SHOULD lm head 守卫的 W4A16 runtime 状态单测，对应 review 要求。
- test/registered/unit/layers/moe/test_flashinfer_cutedsl_dispatch.py（模块 单元测试；类别 test；类型 test-coverage）：修复 main 合并后 CuTe DSL prefill 分派 fixture 漂移：补声明 quant_mode="w4a4"，不改变运行时逻辑。
- docs/docs/references/environment_variables.mdx（模块 文档；类别 docs；类型 documentation）：补充 SGLANG_FLASHINFER_CUTEDSL_NVFP4_W4A16 的环境变量文档。

关键符号：fp4_gemm, ModelOptFp4LinearMethod.process_weights_after_loading, ModelOptFp4LinearMethod.apply, ensure_cutedsl_wrapper, fused_experts_none_to_flashinfer_cutedsl_fp4, fused_experts_flashinfer_to_flashinfer_cutedsl_fp4, refresh_cutedsl_standard_scales_for_weight_update, should_apply_lm_head_quant_method, handle_a2a_moe, _maybe_enable_silu_fp4_quant_fusion

关键源码片段

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

dense 路径核心改造：fp4_gemm 增加 quant_mode 分派、ModelOptFp4LinearMethod 支持 W4A16 权重预打包与 apply 分支，并影响 dispatch 决策。

```
@register_custom_op(fake_impl=_sglang_fp4_gemm_fake)
def fp4_gemm(
    input: torch.Tensor,
    weight: torch.Tensor,
    # W4A16 下激活不量化，input_sf 允许为 None
    input_sf: Optional[torch.Tensor],
    weight_sf: torch.Tensor,
    alpha: torch.Tensor,
    out_dtype: torch.dtype,
    out_features: int,
    # 新增 quant_mode 分派参数，默认 w4a4 保持原路径
    quant_mode: str = "w4a4",
) -> torch.Tensor:
    if not enable_flashinfer_fp4_gemm:
        raise RuntimeError(
            "NVFP4 GEMM requires flashinfer's mm_fp4; please install flashinfer."
        )
    fp4_backend = get_fp4_gemm_runner_backend()
    # 复用 backend 名到 FlashInfer API 名的映射逻辑
    backend = fp4_backend.get_flashinfer_backend()

    if quant_mode == "w4a4":
        # W4A4：激活已量化为 NVFP4，需要 input_sf 参与反量化
        return flashinfer_fp4_gemm(
            input, weight, input_sf, weight_sf, alpha, out_dtype, backend=backend
        )
    elif quant_mode == "w4a16":
        # W4A16：激活保持 BF16，只对权重做 NVFP4 反量化，
        # 因此不传 input_sf，改用 mm_bf16_fp4 入口
        from flashinfer import mm_bf16_fp4

        return mm_bf16_fp4(
            input,
            weight,
            weight_sf,
            alpha,
            backend=backend,
            out_dtype=out_dtype,
        )
    else:
        raise ValueError(f"Unsupported FlashInfer FP4 GEMM quant mode: {quant_mode}")
```

python/sglang/srt/layers/moe/moe_runner/flashinfer_cutedsl.py

MoE 路径核心改造：CuteDslMoEWrapper 传入 quant_mode，fused 函数在 W4A16 下直通 BF16 激活并跳过 FP4 布局处理。

```
    # 激活量化分支：per-token W4A4 / 静态 W4A4 / W4A16 三种模式
    if quant_info.use_per_token_activation:
        # per-token W4A4：动态计算每 token 的激活 scale
```

```
        from flashinfer import SfLayout, nvfp4_quantize

        x_fp4, x_sf, per_token_scale = nvfp4_quantize(
            hidden_states,
            quant_info.a1_scale,
            sfLayout=SfLayout.layout_linear,
            per_token_activation=True,
            backend="cute-dsl",
        )
    elif quant_info.quant_mode == "w4a16":
        # W4A16：BF16 激活直通，跳过 NVFP4 量化，
        # 也不构造 x_sf 与 per_token_scale
        x_fp4 = hidden_states
        x_sf = None
        per_token_scale = None
    else:
        # 静态 W4A4：使用 checkpoint 的 a1_scale 做 block 量化
        x_fp4, x_sf = fp4_quantize(
            hidden_states,
            quant_info.a1_scale,
            sf_vec_size=_FP4_SF_VEC_SIZE,
            is_sf_swizzled_layout=False,
        )
        per_token_scale = None

    # W4A16 下 x_fp4 就是 BF16 激活，不需要 reshape 成
    # [seq_len, hidden_size // 2] 的打包 FP4 布局
    if quant_info.quant_mode != "w4a16":
        seq_len, hidden_size = hidden_states.shape
        x_fp4 = x_fp4.reshape(seq_len, hidden_size // 2)
        x_sf = x_sf.view(torch.float8_e4m3fn).reshape(
            seq_len, hidden_size // _FP4_SF_VEC_SIZE
        )

    output = quant_info.wrapper.run(
        x=x_fp4,
        x_sf=x_sf,
        token_selected_experts=topk_ids,
        token_final_scales=topk_weights,
        w1_weight=quant_info.w13_weight,
        w1_weight_sf=quant_info.w13_weight_sf,
        w1_alpha=quant_info.w1_alpha,
        # W4A16 下 GEMM2 不需要激活量化 scale
        fc2_input_scale=(
            None if quant_info.quant_mode == "w4a16" else quant_info.a2_scale
        ),
        w2_weight=quant_info.w2_weight,
        w2_weight_sf=quant_info.w2_weight_sf,
        w2_alpha=quant_info.w2_alpha,
```

```
        per_token_scale=per_token_scale,
    )
```

评论区精华

Review 讨论集中在四点上：

- qwen3_5.py 需要同款融合守卫：mmangkad 在 deepseek_v2.py 的 diff 上指出 qwen3_5.py 的 _maybe_enable_silu_fp4_quant_fusion 需要同样的 quant_mode == "w4a4" 守卫。zianglih 确认这是 W4A4-only 融合，已在提交 42c43829ee 补上。
- env 未设置时的误触发风险：mmangkad 询问 moe_hook.py 中 SGLANG_MOE_NVFP4_DISPATCH.get() 在 env 未设置时是否会错误抛错。zianglih 回复：未设置时该检查为 false 不会抛错，但会落入下方自动启用逻辑——已在 42c43829ee 修复，W4A16 下保持 dispatch 禁用，只有显式 true 才失败。
- dispatcher workspace 是否跟随该开关：mmangkad 建议 dispatcher workspace 的尺寸选择也应跟随 W4A16 标志。zianglih 认为修复 resolution 后 dispatch 保持 false，现有 sizing 已按 BF16 几何选择，无需第二次 workspace override。
- LM head 守卫测试覆盖：mmangkad 要求 should_apply_lm_head_quant_method 的新分支在 test_modelopt_loader.py 中有测试，zianglih 在 42c43829ee 中补充了 test_lm_head_guard_accepts_modelopt_fp4_cutedsl_w4a16_runtime_state。
  - qwen3_5.py 需要同样的 W4A4 融合守卫 (correctness): zianglih 确认 fusion 是 W4A4-only，已在提交 42c43829ee 中为 qwen3_5.py 补上同款守卫。
  - moe_hook 中 env 未设置时是否会误触发 (correctness): 已在 42c43829ee 修复：W4A16 下保持 dispatch 禁用，只有显式设置 true 才抛 ValueError。
  - dispatcher workspace 是否应跟随 W4A16 标志 (design): 讨论后未增加额外 override，依赖 dispatch=false 的 BF16 几何选择。
  - LM head 量化守卫需要测试覆盖 (testing): zianglih 在 42c43829ee 中新增 test_lm_head_guard_accepts_modelopt_fp4_cutedsl_w4a16_runtime_state。

风险与影响

- 风险：
  - FlashInfer 版本硬绑定：PR body 明确声明“no compatibility fallback for earlier FlashInfer APIs”，若运行环境回退到 0.6.18 之前，mm_bf16_fp4、prepare_bf16_fp4_weights、CuteDslMoEWrapper(quant_mode=...) 等调用会直接 ImportError 或参数错误，装机时必须锁定三件套版本。
  - 核心 GEMM 路径契约变更：fp4_gemm 的 input_sf 由必填改为 Optional 并新增 quant_mode 参数，所有调用点虽经默认值保持原行为，但未来新增调用方容易漏传 quant_mode 导致静默走 W4A4。
  - 权重重载与 CUDA graph 绑定：prepare_bf16_fp4_weights 产出的 weight 若在 reload 时 shape/dtype/device 变化，refresh_cutedsl_standard_scales_for_weight_update 会抛 RuntimeError 要求 recapture；测试只覆盖了同权重路径，换权重场景未覆盖。

- 配置组合脆弱：W4A16 与 SGLANG_MOE_NVFP4_DISPATCH=1、DeepEP masked A2A 的组合在启动期直接 raise，用户需要精确匹配 env 组合，误配置反馈在启动时而不是运行时，对集群管理员友好但缺省无引导。
- 双路径长期维护：w4a4/w4a16 分支散布于 quant、MoE runner、dispatcher、logits_processor、模型定义五处，后续 FlashInfer API 演进或新增融合 kernel 时容易漏掉某个分支的守卫。

- 影响：
  - 用户影响：功能默认关闭，对现有 W4A4 用户零影响；B300/GB300 上追求精度的用户可通过单个环境变量启用 W4A16，MoE 与 dense 层同时生效。
  - 系统影响：MoE 与 dense 双路径、LM head、A2A 两种模式（无 A2A / FlashInfer A2A）均纳入支持，覆盖 online 量化与 serialized ModelOpt 两类 NVFP4 权重来源。
  - 团队影响：FlashInfer 升级节奏被绑定到 0.6.18+；后续维护需在两组 quant_mode 分支间保持同步，测试矩阵新增 nightly 与 extra-b 两个 4-gpu-b200 用例，CI 时长增加。
  - 风险标记：FlashInfer 版本硬绑定无 fallback，核心 GEMM 路径契约变更，双 quant_mode 分支维护成本，Blackwell 专属功能，CUDA graph 与权重重载绑定

关联脉络

- PR #36954 [Fridge003] Bump FlashInfer to 0.6.18: 本 PR 依赖 FlashInfer v0.6.18 的 CuTe DSL NVFP4 MoE W4A16、SiTU、workspace-lifetime 与 SM100/SM103 dense W4A16 API，PR body 明确要求先合入该钉版 PR。