PR #25655 完整报告

sgl-project/sglang

Feat/add w4a16 moe support to nemotron

合并时间：2026-06-03 13:42

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/25655

执行摘要

- 一句话：支持 Nemotron 模型 NVFP4 权重通过 Marlin W4A16 在 SM80-SM90 上推理
- 推荐动作：建议精读：该 PR 展示了如何将专有量化格式（NVFP4 ModelOpt）映射到已有 Marlin 内核，包含 scale 转换、非门控 MoE 扩展、多后端路由等设计决策，对于理解 SGLang 的量化抽象层和 MoE 支持有参考价值。关注点：scale 转换的数值正确性、非门控 MoE 的激活函数处理、全局 scale 指数偏移的数学推导。

功能与动机

Serve NVFP4 modelopt checkpoints (modelopt_fp4) on Ampere/Hopper by routing them through Marlin W4A16 when native FP4 is not supported. （PR Body 原文）

实现拆解

1. NVFP4 到 Marlin 的 scale 适配层：在 marlin_utils_fp4.py 中新增 nvfp4_marlin_process_scales 将 per-group scale 从 FP16/BF16 转换为 Marlin 所需的 FP8E4M3 格式；新增 nvfp4_marlin_process_global_scale 将全局 scale 进行指数偏移以匹配内核期望；新增 apply_fp4_marlin_linear 作为主入口，通过 gptq_marlin_gemm 调用 Marlin 内核，支持 FP16/BF16 激活；通过 register_custom_op 注册 fake 实现用于 tracing。
2. Dense Linear 集成：在 modelopt_quant.py 的 ModelOptNvFp4LinearMethod 中，create_weights 保存 quant_config 和 params_dtype，process_weights_after_loading 检测当前后端是否为 Marlin，若是则调用 prepare_nvfp4_layer_for_marlin 执行权重和 scale 的 repack，并跳过原生的 Blackwell 路径；apply 方法也优先路由到 apply_fp4_marlin_linear。同时 get_min_capability 从 100 降至 80 以允许 SM80+ 运行。
3. MoE 扩展：fused_marlin_moe.py 中扩展 get_scalar_type 接受 global_scale 参数以匹配 NVFP4 的 scale 语义；新增 is_nvfp4_marlin 检测逻辑；fused_marlin_moe 函数增加 w1_global_scale、w2_global_scale、activation、is_gated 参数，并据此计算 gemm1_n 和控制激活函数的使用（支持 silu 和 relu2）。MarlinMoERunner 在 moe_runner/marlin.py 中传递全局 scale。
4. FP4 GEMM 后端选择：fp4_utils.py 的 Fp4GemmRunnerBackend 新增 MARLIN；initialize_fp4_gemm_config 在 SM80-SM90 且 CUDA 时自动选择 marlin，并优先于其他后端（flashinfer 等仅限 SM100+ 或 CPU）。

5. 模型钩子：nemotron_h_hook.py 在检测到 modelopt_fp4 或 modelopt_mixed 量化且 SM80-SM90 时，自动设置 --moe-runner-backend marlin，避免用户手动指定。
6. 测试与文档：新增单元测试验证 scale 变换数值正确性（test_gptq_marlin.py）、dense linear 与 dequant 参考的精度（test_gptq_marlin.py）、MoE 非门控 W4A16 和 NVFP4 的端到端精度（test_moe_wna16_marlin.py）。新增端到端模型测试 test_nvidia_nemotron_3_super_nvfp4.py。更新 docs_new/docs/references/fp4_gemm_backend.md 和 server_args 文档。

关键文件：

- python/sglang/srt/layers/quantization/marlin_utils_fp4.py（模块 量化层；类别 source；类型 core-logic；符号 nvfp4_marlin_process_scales, nvfp4_marlin_process_global_scale, fake_apply_fp4_marlin_linear, apply_fp4_marlin_linear）：核心新增文件，实现 NVFP4 到 Marlin 的 scale 转换、dense linear apply 函数和权重 prep 逻辑。所有 NVFP4 专用内核适配均在此完成。
- python/sglang/srt/layers/quantization/modelopt_quant.py（模块 量化配置；类别 source；类型 data-contract；符号 create_weights, process_weights_after_loading, apply）：Dense layer 集成入口，加载权重时路由到 Marlin 准备，apply 时调用 Marlin GEMM。同时降低最低计算能力限制，使 NVFP4 量化的模型可在 SM80+ 上加载。
- python/sglang/srt/layers/moe/fused_moe_triton/fused_marlin_moe.py（模块 MoE 内核；类别 source；类型 core-logic；符号 get_scalar_type, fused_marlin_moe）：MoE 核心函数扩展，新增 NVFP4 检测、全局 scale 支持、非门控激活函数。使 fused_marlin_moe 能够处理 NVFP4 MoE 层。
- python/sglang/jit_kernel/tests/test_moe_wna16_marlin.py（模块 MoE 测试；类别 test；类型 test-coverage；符号 test_fused_marlin_moe_non_gated_relu2, test_fused_marlin_moe_nvfp4_non_gated_padded_intermediate_launches, test_fused_marlin_moe_nvfp4_non_gated_matches_dequant_reference）：新增 MoE 非门控 NVFP4 和 W4A16 的单元测试，覆盖 fused_marlin_moe 的 non-gated 路径和精度比对。
- python/sglang/jit_kernel/tests/test_gptq_marlin.py（模块 内核测试；类别 test；类型 test-coverage；符号 test_nvfp4_marlin_support_and_scale_transforms_sm80_sm90, test_nvfp4_marlin_dense_matches_dequant_reference）：新增 NVFP4 scale 变换和 dense linear 精度测试，验证 scale 处理正确性以及 apply_fp4_marlin_linear 与 dequant 参考的一致性。
- python/sglang/srt/layers/quantization/fp4_utils.py（模块 后端选择；类别 source；类型 core-logic；符号 Fp4GemmRunnerBackend, initialize_fp4_gemm_config）：FP4 后端枚举增加 MARLIN，并在初始化函数中为 SM80-SM90 自动选择 Marlin 后端，影响所有 FP4 量化模型的推理路径。
- python/sglang/srt/arg_groups/nemotron_h_hook.py（模块 模型钩子；类别 source；类型 dependency-wiring；符号 apply_nemotron_h_defaults）：模型钩子，针对 Nemotron NVFP4 模型自动设置 MoE runner 为 marlin，简化用户配置。
- python/sglang/srt/layers/moe/moe_runner/marlin.py（模块 MoE 执行器；类别 source；类型 core-logic；符号 MarlinMoERunner）：MoE runner 增加对 global_scale 参数的传

递，确保 NVFP4 全局 scale 到达 kernel。

- python/sglang/srt/layers/quantization/marlin_utils.py（模块 量化工具；类别 source；类型 core-logic；符号 MARLIN_SUPPORTED_GROUP_SIZES, check_marlin_supported）：修改全局 MARLIN_SUPPORTED_GROUP_SIZES，加入 16 以支持 NVFP4 的 group_size=16，同时新增 check 逻辑避免影响其他量化类型。
- python/sglang/jit_kernel/utils.py（模块 JIT 工具；类别 source；类型 core-logic；符号 _local_jit_source_hash）：改进 JIT 缓存哈希机制，使其包含源代码文件内容，确保重新编译的正确性。减少因头文件变更导致的缓存未命中问题。
- python/sglang/test/test_marlin_utils.py（模块 测试工具；类别 test；类型 test-coverage；符号 make_nvfp4_weight_and_ref, _unpack）：新增 make_nvfp4_weight_and_ref 辅助函数，为 NVFP4 单元测试提供构造 NVFP4 权重和 dequant 参考的函数。
- python/sglang/jit_kernel/csrc/gemm/marlin/marlin_template.h（模块 内核模板；类别 source；类型 core-logic）：Marlin 内核模板微调，可能涉及 NVFP4 相关的配置，确保 JIT 编译通过。

关键符号：nvfp4_marlin_process_scales, nvfp4_marlin_process_global_scale, apply_fp4_marlin_linear, prepare_nvfp4_layer_for_marlin, prepare_moe_nvfp4_layer_for_marlin, fused_marlin_moe, get_scalar_type, check_marlin_supported

关键源码片段

python/sglang/srt/layers/quantization/marlin_utils_fp4.py

核心新增文件，实现 NVFP4 到 Marlin 的 scale 转换、dense linear apply 函数和权重 prep 逻辑。所有 NVFP4 专用内核适配均在此完成。

```
# python/sglang/srt/layers/quantization/marlin_utils_fp4.py

def nvfp4_marlin_process_scales(marlin_scales: torch.Tensor) -> torch.Tensor:
    # NVFP4 ModelOpt scales 应为非负值，但保留警告以便诊断异常 checkpoint
    if not (marlin_scales >= 0).all():
        import logging
        logging.getLogger(__name__).warning_once(
            "NVFP4 Marlin assumes non-negative scales, but negative scales "
            "were found. Accuracy may be degraded."
        )

    # 转换为 FP16 后执行通道顺序重排 (0,2,1,3)
    marlin_scales = marlin_scales.to(torch.half)
    marlin_scales = marlin_scales.view(-1, 4)[:, [0, 2, 1, 3]].view(
        marlin_scales.size(0), -1
    )
    # 乘以 2^7 并左移 1 位，以 FP16 位模式表达 FP8E4M3 指数
    marlin_scales = (marlin_scales * (2**7)).view(torch.int16) << 1
    marlin_scales = marlin_scales.view(torch.float8_e4m3fn)
    # Marlin kernel 仅消费偶数索引的 scale
```

```python
    return marlin_scales[:, 1::2].contiguous()


@register_custom_op(fake_impl=fake_apply_fp4_marlin_linear)
def apply_fp4_marlin_linear(
    input: torch.Tensor,
    weight: torch.Tensor,
    weight_scale: torch.Tensor,
    weight_global_scale: torch.Tensor,
    workspace: torch.Tensor,
    size_n: int,
    size_k: int,
    bias: torch.Tensor | None = None,
    use_fp32_reduce: bool = USE_FP32_REDUCE_DEFAULT,
) -> torch.Tensor:
    if input.dtype not in (torch.float16, torch.bfloat16):
        raise RuntimeError("NVFP4 Marlin requires FP16 or BF16 activations.")

    reshaped_x = input.reshape(-1, input.shape[-1])
    out_shape = input.shape[:-1] + (size_n,)

    use_atomic_add = should_use_atomic_add_reduce(
        m=reshaped_x.size(0), n=size_n, k=size_k,
        device=input.device, dtype=input.dtype,
    )

    # 调用 JIT 编译的 Marlin GEMM kernel，传入全局 scale 参数
    output = gptq_marlin_gemm(
        a=reshaped_x,
        c=None,
        b_q_weight=weight,
        b_scales=weight_scale,
        global_scale=weight_global_scale,
        b_zeros=None,
        g_idx=None,
        perm=None,
        workspace=workspace,
        b_q_type=scalar_types.float4_e2m1f,
        size_m=reshaped_x.size(0),
        size_n=size_n,
        size_k=size_k,
        is_k_full=True,
        use_atomic_add=use_atomic_add,
        use_fp32_reduce=use_fp32_reduce,
    )

    if bias is not None:
        output.add_(bias)
```

```
    return output.reshape(out_shape)
```

python/sglang/srt/layers/moe/fused_moe_triton/fused_marlin_moe.py

MoE 核心函数扩展，新增 NVFP4 检测、全局 scale 支持、非门控激活函数。使 fused_marlin_moe 能够处理 NVFP4 MoE 层。

```
# python/sglang/srt/layers/moe/fused_moe_triton/fused_marlin_moe.py

def get_scalar_type(
    num_bits: int,
    has_zp: bool,
    scales: Optional[torch.Tensor] = None,
    global_scale: Optional[torch.Tensor] = None,
):
    from sgl_kernel.scalar_type import scalar_types

    # NVFP4 通过 global_scale 区分：若有 global_scale 则认为是 float4_e2m1f
    if not has_zp and num_bits == 4 and scales is not None and (
        scales.dtype == torch.float8_e8m0fnu or global_scale is not None
    ):
        return scalar_types.float4_e2m1f
    if has_zp:
        assert num_bits == 4
        return scalar_types.uint4
    else:
        return scalar_types.uint4b8 if num_bits == 4 else scalar_types.uint8b128


# 在 fused_marlin_moe 函数内部检测 NVFP4
is_nvfp4_marlin = (
    num_bits == 4
    and w1_zeros is None
    and w2_zeros is None
    and w1_global_scale is not None
    and w2_global_scale is not None
)

if is_mxfp4_marlin:
    assert hidden_states.dtype == torch.bfloat16, ...
elif not is_nvfp4_marlin:
    assert hidden_states.dtype == w1_scale.dtype, ...

# 非门控时 gemm1_n = N 而不是 2*N
gemm1_n = 2 * N if is_gated else N
```

评论区精华

全局 group size 修改风险

TomerBN-Nvidia 指出在 MARLIN_SUPPORTED_GROUP_SIZES 中加入 16 会影响所有 Marlin 量化类型的 check_marlin_supported，建议添加 bypass。shaunkotek 随后在 check_marlin_supported 中增加了对 group_size 的额外验证，仅在 NVFP4 对应的 float4_e2m1f 时允许 16。

FP4 后端范围限制

TomerBN-Nvidia 提醒 initialize_fp4_gemm_config 中的 elif 条件会误匹配 SM10+/11+ 设备到 Marlin，shaunkotek 修复为 (8, 0) <= capability < (10, 0)。

MoE 非门控断言

TomerBN-Nvidia 建议将 is_nvfp4_marlin 加入 is_mxfp4_marlin 的断言条件。shaunkotek 解释 mxfp4 只支持 BF16、nvfp4 支持 FP16 和 BF16，因此保留分支判断，最终 TomerBN 认可。

测试覆盖建议

b8zhong 要求添加端到端模型测试，shaunkotek 新增了 test_nvidia_nemotron_3_super_nvfp4.py。

代码复用

b8zhong 建议在测试中使用已有的 is_sm90_supported / is_sm80_supported 替代自定义函数，shaunkotek 修改。

- 全局 group size 修改风险 (correctness): shaunkotek 随后在 check_marlin_supported 中增加了对 group_size 的额外验证，仅在 NVFP4 对应的 float4_e2m1f 时允许 16。
- FP4 后端范围限制 (correctness): shaunkotek 修复为 (8, 0) <= capability < (10, 0)。
- MoE 非门控断言 (correctness): 最终 TomerBN 认可当前设计，保留独立分支。
- 测试覆盖建议 (testing): shaunkotek 新增了 test_nvidia_nemotron_3_super_nvfp4.py。

风险与影响

- 风险:
  - 数值精度风险：NVFP4 到 Marlin W4A16 的 scale 转换涉及指数偏移和浮点 reinterpret，可能引入精度损失。已有 dense linear 和 MoE 单元测试，但 CI 中由于 JIT 编译时间过长被 skip，需手动运行验证。
  - 全局影响：MARLIN_SUPPORTED_GROUP_SIZES 的修改经 bypass 后影响受控，但仍需警惕其他量化类型意外接受 group size 16。
  - 兼容性：NVFP4 路径仅在 SM80-SM90 启用，SM100+ 仍使用原生 FP4 后端，不会影响已有功能。
  - 性能：Marlin W4A16 相比原生 FP4 可能略慢，但这是必要的 fallback。
- 影响:
  - 用户： Nemotron NVFP4 模型现在可以在 A100/H100 上推理，需指定 --fp4-gemm-backend marlin（或由 nemotron hook 自动选择）。
  - 系统：新增 Marlin 后端，不影响现有 flashinfer/cutlass 后端选择逻辑。

- 团队：需维护 NVFP4 适配层和测试，CI 中包括端到端模型测试（但被 skip，待解决编译时间问题）。
- 风险标记：核心量化路径变更，测试在 CI 中被 skip，全局 group size 兼容性

关联脉络

- 暂无明显关联 PR