PR #24986 完整报告

sgl-project/sglang

[rebase]Deepseek_v4 support w4(mxfp4)a16 on hopper

合并时间：2026-05-14 07:33

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/24986

执行摘要

- 一句话：DeepSeek V4 新增 Hopper MXFP4 Marlin 支持
- 推荐动作：该 PR 是对 DeepSeek V4 MXFP4 量化支持的关键移植，值得关注其权重名称兼容性设计和 Marlin 集成模式。建议团队统一量化体系结构后考虑合并两条后端。

功能与动机

将 deepseek_v4 开发分支中的 MXFP4 量化支持合并到主分支，使得 DeepSeek V4 模型能够在 Hopper (SM90) GPU 上使用基于 Marlin 的 W4A16 推理。该功能最初由 #23686 实现，本次 PR 通过 rebase 方式移植。

实现拆解

1. 灵活的权重参数获取：在 marlin_utils_fp4.py 中添加 _get_optional_param 函数，支持从 layer 中按多个候选名称获取参数（如优先 w13_weight_scale 再 fallback 到 w13_weight_scale_inv），以兼容不同 checkpoint 命名。
2. Marlin MoE 权重初始化：在 mxfp4_marlin_moe.py 中重写 create_weights 方法，直接创建 int8 类型的量化权重和 float32 类型的 scale 参数（block 大小 32，标记 format_ue8m0=False），不再依赖底层的 FP8 方法。
3. MXFP4 Marlin 路径集成：在 mxfp4.py 的 Mxfp4MoEMethod 中新增 use_marlin 标志，在 process_weights_after_loading 中优先执行 Marlin 预处理（调用 prepare_moe_mxfp4_layer_for_marlin），在 apply 中构造 MarlinMoeQuantInfo 并执行 Marlin 推理。
4. Marlin 内核断言调整：在 fused_marlin_moe.py 中，将 MXFP4 模式的 dtype 检查从检查 scale 类型改为断言激活类型必须为 bfloat16；在 moe_wna16_marlin.cuh 中增加对 float4_e2m1f 量化类型的运行时检查，要求 group_size 为 16 或 32，且 group_size=32 时激活必须为 bfloat16。
5. CI 稳定性增强：在 DSV4 Flash FP4/FP8 测试中增加 --watchdog-timeout 900 参数，避免因 nvshmem 偶发错误导致 watchdog 误杀。

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 _get_optional_param）：核心量化预处理函数，新增 _get_optional_param 实现多名称参数获取，重构 prepare_moe_mxfp4_layer_for_marlin

以兼容新旧命名。

- python/sglang/srt/layers/quantization/mxfp4_marlin_moe.py（模块 量化层；类别 source；类型 dependency-wiring）：Marlin MoE 后端核心实现，新增独立的 create_weights 方法，直接创建 int8 权重和 float32 scales。
- python/sglang/srt/layers/quantization/mxfp4.py（模块 量化层；类别 source；类型 dependency-wiring）：Marlin 路径集成入口，新增 use_marlin 标志并在 process_weights_after_loading 和 apply 中路由。
- python/sglang/srt/layers/moe/fused_moe_triton/fused_marlin_moe.py（模块 Marlin 核；类别 source；类型 core-logic）：Marlin 内核入口，调整 MXFP4 模式的 dtype 断言。
- python/sglang/jit_kernel/csrc/gemm/marlin_moe/moe_wna16_marlin.cuh（模块 Marlin 核；类别 source；类型 core-logic）：CUDA 核函数运行时检查，新增对 MXFP4 量化类型的约束。
- test/registered/dsv4/test_deepseek_v4_flash_fp4_h200.py（模块 测试；类别 test；类型 test-coverage）：DSV4 Flash FP4 测试，增加 watchdog 超时以避免 CI 不稳定。
- test/registered/dsv4/test_deepseek_v4_flash_fp8_h200.py（模块 测试；类别 test；类型 test-coverage）：DSV4 Flash FP8 测试，同步增加 watchdog 超时。

关键符号：marlin_utils_fp4._get_optional_param, marlin_utils_fp4.prepare_moe_mxfp4_layer_for_marlin, Mxfp4MarlinMoEMethod.create_weights, Mxfp4MoEMethod.process_weights_after_loading, Mxfp4MoEMethod.apply, fused_marlin_moe, moe_wna16_marlin_gemm

关键源码片段

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

核心量化预处理函数，新增 _get_optional_param 实现多名称参数获取，重构 prepare_moe_mxfp4_layer_for_marlin 以兼容新旧命名。

```
def _get_optional_param(layer: torch.nn.Module, *names: str) -> torch.Tensor | None:
    # 按顺序尝试多个属性名，返回第一个非 None 的值。
    # 用于兼容不同 checkpoint 格式（旧命名 vs 新命名）
    for name in names:
        value = getattr(layer, name, None)
        if value is not None:
            return value
    return None


def prepare_moe_mxfp4_layer_for_marlin(layer: torch.nn.Module) -> None:
    group_size = 32
    w13 = layer.w13_weight.data
    w2 = layer.w2_weight.data
    # 支持 w13_weight_scale 和 w13_weight_scale_inv 两种命名
    w13_scale = _get_optional_param(
        layer, "w13_weight_scale", "w13_weight_scale_inv"
    )
```

```
    w2_scale = _get_optional_param(
        layer, "w2_weight_scale", "w2_weight_scale_inv"
    )
    w13_bias = _get_optional_param(
        layer, "w13_weight_bias", "w13_bias"
    )
    w2_bias = _get_optional_param(
        layer, "w2_weight_bias", "w2_bias"
    )

    if w13_scale is None or w2_scale is None:
        raise ValueError("MXFP4 Marlin requires w13/w2 weight scales.")

    # 提取底层 data，兼容 Parameter 和普通 Tensor
    w13_scale_data = (
        w13_scale.data if hasattr(w13_scale, "data") else w13_scale
    )
    w2_scale_data = (
        w2_scale.data if hasattr(w2_scale, "data") else w2_scale
    )
    # ... 后续使用 data 进行重排和注册
    # 注意：最终注册的参数名统一为 w13_weight_scale / w2_weight_scale
```

python/sglang/srt/layers/quantization/mxfp4_marlin_moe.py

Marlin MoE 后端核心实现，新增独立的 create_weights 方法，直接创建 int8 权重和 float32 scales。

```
  def create_weights(
      self,
      layer: Module,
      num_experts: int,
      hidden_size: int,
      intermediate_size_per_partition: int,
      params_dtype: torch.dtype,
      **extra_weight_attrs,
  ):
      from sglang.srt.layers.moe.fused_moe_triton import (
          FusedMoeWeightScaleSupported,
      )

      fp4_block_k = 32

      # int8 量化权重，shape: (E, N, K//2)
      w13_weight = torch.nn.Parameter(
          torch.empty(
              num_experts,
              2 * intermediate_size_per_partition,
              hidden_size // 2,
              dtype=torch.int8,
```

```
        ),
        requires_grad=False,
    )
    w2_weight = torch.nn.Parameter(
        torch.empty(
            num_experts,
            hidden_size,
            intermediate_size_per_partition // 2,
            dtype=torch.int8,
        ),
        requires_grad=False,
    )
    layer.register_parameter("w13_weight", w13_weight)
    layer.register_parameter("w2_weight", w2_weight)
    set_weight_attrs(w13_weight, extra_weight_attrs)
    set_weight_attrs(w2_weight, extra_weight_attrs)

    # float32 scale, block size 32, 不采用 UE8M0 格式
    w13_weight_scale = torch.nn.Parameter(
        torch.ones(
            num_experts,
            2 * intermediate_size_per_partition,
            hidden_size // fp4_block_k,
            dtype=torch.float32,
        ),
        requires_grad=False,
    )
    w2_weight_scale = torch.nn.Parameter(
        torch.ones(
            num_experts,
            hidden_size,
            intermediate_size_per_partition // fp4_block_k,
            dtype=torch.float32,
        ),
        requires_grad=False,
    )
    w13_weight_scale.format_ue8m0 = False
    w2_weight_scale.format_ue8m0 = False
    scale_attrs = dict(extra_weight_attrs)
    scale_attrs["quant_method"] = FusedMoeWeightScaleSupported.BLOCK.value
    layer.register_parameter("w13_weight_scale_inv", w13_weight_scale)
    layer.register_parameter("w2_weight_scale_inv", w2_weight_scale)
    set_weight_attrs(w13_weight_scale, scale_attrs)
    set_weight_attrs(w2_weight_scale, scale_attrs)
```

评论区精华

CI 运行中暴露出 nvshmem 错误导致 watchdog 触发的问题。作者 @shiyu7 分析后指出 nvshmem 错误与 watchdog 生成 dump 相关，并建议增加 watchdog 超时到 900s。

Reviewer @Fridge003 接受了该方案并 re-run CI，最终测试通过。未出现其他设计层面的分歧。

- CI 稳定性：nvshmem 错误导致 watchdog 触发 (other): 通过增加 --watchdog-timeout 900 解决，CI 重新运行后通过。

风险与影响

- 风险：
  - 兼容性风险：scale 参数名称从 w13_weight_scale_inv 改为 w13_weight_scale，旧 checkpoint 可能加载失败（虽有 fallback 但仅在 _get_optional_param 中处理，需确保所有调用点都使用该函数）。
  - 断言变更：fused_marlin_moe 中强制要求 MXFP4 时激活为 bfloat16，可能不兼容 fp16 激活的场景（但 Marlin MoE 之前也隐含该要求）。
  - 路径冲突：在 mxfp4.py 中新增的 Marlin 路径处理位于 process_weights_after_loading 开头，通过 return 提前退出，需保证不与其他后端（FlashInfer、Triton）的后续逻辑相干扰。
  - 硬件限制：Marlin 核仅在 Hopper (SM90) 上可用，非 Hopper 设备会抛出 RuntimeError，用户需明确知晓。
- 影响：
  - 用户：DeepSeek V4 模型用户可以通过 --moe-runner-backend marlin 选择 MXFP4 Marlin 后端，获得 W4A16 推理能力。
  - 系统：新增约 150 行核心代码，主要影响量化层和 MoE runner 的选择逻辑。
  - 团队：需要维护两条 MXFP4 后端路径（FlashInfer 和 Marlin），增加测试和兼容性负担。
  - 风险标记：权重命名兼容性风险，断言强制 bfloat16 激活，仅 Hopper GPU 支持，CI 可靠性依赖超时调整

关联脉络

- PR #23686 原始 MXFP4 支持 PR（参考）：当前 PR 的 body 明确引用 #23686 作为原始实现，本次是 rebase 移植。
- PR #24816 Add FlashInfer SM90 cutlass MXFP4 MoE backend: 同为 DeepSeek V4 的 MXFP4 量化后端，但使用不同 MoE runner，两者构成互补路径。