PR #31989 完整报告

sgl-project/sglang

feat: Support nvidia/MiniMax-M3-NVFP4

合并时间：2026-07-31 05:32

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/31989

执行摘要

- 一句话：支持 MiniMax-M3 NVFP4 混合精度模型加载
- 推荐动作：建议读者关注 ModelOptMixedPrecisionConfig 的扩展方式以及如何通过 hf_to_sglang_mapper 和 packed_modules_mapping 解决权重名称映射问题。这种设计模式值得在其他需要适配不同前缀的量化模型中复用。

功能与动机

此 PR 修复 Issue #31827，该问题报告 nvidia/MiniMax-M3-NVFP4 模型在加载时崩溃，原因是 fused MoE 回退到未量化方法，以及 MXFP8 线性层权重被静默丢弃。通过正确处理混合精度量化配置，使该模型能够正常加载和推理。

实现拆解

1. 在 modelopt_quant.py 的 ModelOptMixedPrecisionConfig 中添加 mxfp8_config 属性，并在 from_config 中从 hf_quant_config 的 quantized_layers 提取 MXFP8 配置；get_quant_method 中新增对 "MXFP8" 的处理，返回 Fp8LinearMethod（线性层）和 Fp8MoEMethod（MoE 层）。
2. 在 minimax_m3.py 和 minimax_m3_vl.py 的模型类中添加 hf_to_sglang_mapper 类属性，将 HF 权重前缀 block_sparse_moe 映射为 SGLang 的 mlp，使量化配置在运行时能正确匹配路由专家层。
3. 在上述两个模型的 determine_num_fused_shared_experts 方法中增加对 modelopt_mixed 量化配置的检测，当共享专家和路由专家使用不同量化格式时禁用共享专家融合，避免计算错误。
4. 在 overrides.py 的 _minimax_m3_overrides 中为 MiniMax-M3 添加默认 MoE 后端选择，当量化方法为 modelopt_mixed 时自动选择 flashinfer_trtllm_routed。
5. 在 flashinfer_trtllm.py 的 FlashInferTrtllmFp4MoeQuantInfo 中添加 gemm1_alpha 和 gemm1_beta 属性，并在 fused_experts_none_to_flashinfer_trtllm_fp4 函数中将其传给底层 kernel，支持 NVFP4 的 alpha/beta 缩放。
6. 新增单元测试 test_minimax_mixed_precision_resolves_runtime_names_and_mxfp8，验证权重名称映射、量化算法解析以及 MXFP8 方法分发是否正确。

关键文件：

- python/sglang/srt/layers/quantization/modelopt_quant.py（模块 量化配置；类别 source；类型 data-contract；符号 ModelOptMixedPrecisionConfig, from_config, get_quant_method, Fp8LinearMethod）：核心配置变更：添加 mxfp8_config 支持 MXFP8 层，扩展 get_quant_method 分发逻辑
- python/sglang/srt/models/minimax_m3.py（模块 模型定义；类别 source；类型 data-contract；符号 MiniMaxM3SparseForCausalLM.hf_to_sglang_mapper, MiniMaxM3SparseForCausalLM.packed_modules_mapping, determine_num_fused_shared_experts）：添加 hf_to_sglang_mapper 解决前缀映射，并禁用混合精度下的共享专家融合
- python/sglang/srt/models/minimax_m3_vl.py（模块 模型定义；类别 source；类型 data-contract；符号 MiniMaxM3SparseForConditionalGeneration.hf_to_sglang_mapper, MiniMaxM3SparseForConditionalGeneration.packed_modules_mapping, _determine_num_fused_shared_experts）：与文本模型类似的修改，为视觉版本添加前缀映射和混合精度检查
- test/registered/unit/model_loader/test_modelopt_loader.py（模块 测试加载；类别 test；类型 test-coverage；符号 test_minimax_mixed_precision_resolves_runtime_names_and_mxfp8）：新增测试验证权重映射、量化算法解析和 MXFP8 方法分发
- python/sglang/srt/layers/moe/moe_runner/flashinfer_trtllm.py（模块 MoE 运行器；类别 source；类型 core-logic；符号 FlashInferTrtllmFp4MoeQuantInfo.gemm1_alpha, FlashInferTrtllmFp4MoeQuantInfo.gemm1_beta, fused_experts_none_to_flashinfer_trtllm_fp4）：添加 gemm1_alpha 和 gemm1_beta 支持，使 NVFP4 experts 在 flashinfer trtllm 后端正确运行
- python/sglang/srt/arg_groups/overrides.py（模块 参数覆盖；类别 source；类型 core-logic；符号 _minimax_m3_overrides）：为 MiniMax-M3 添加默认 MoE 后端选择，modelopt_mixed 时启用 flashinfer_trtllm_routed

关键符号：ModelOptMixedPrecisionConfig.init, ModelOptMixedPrecisionConfig.from_config, ModelOptMixedPrecisionConfig.get_quant_method, MiniMaxM3SparseForCausalLM.determine_num_fused_shared_experts, MiniMaxM3SparseForConditionalGeneration._determine_num_fused_shared_experts, FlashInferTrtllmFp4MoeQuantInfo.init, fused_experts_none_to_flashinfer_trtllm_fp4, _minimax_m3_overrides

关键源码片段

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

核心配置变更：添加 mxfp8_config 支持 MXFP8 层，扩展 get_quant_method 分发逻辑

```
class ModelOptMixedPrecisionConfig(ModelOptQuantConfig):
    """Configuration for ModelOpt MIXED_PRECISION checkpoints."""

    def __init__(
        self,
        kv_cache_quant_algo: Optional[str],
```

```
        exclude_modules: Optional[List[str]],
        packed_modules_mapping: Optional[Dict[str, List[str]]],
        quantized_layers: Dict[str, Dict[str, Any]],
        fp8_config: ModelOptFp8Config,
        nvfp4_config: ModelOptFp4Config,
        nvfp4a16_config: ModelOptFp4Config,
        mxfp8_config: Fp8Config, # <-- 新增：存储 MXFP8 配置
    ) -> None:
        super().__init__(kv_cache_quant_algo, exclude_modules, packed_modules_mapping)
        self.quantized_layers = quantized_layers
        self.fp8_config = fp8_config
        self.mxfp8_config = mxfp8_config # <-- 新增
        self.nvfp4_config = nvfp4_config
        self.nvfp4a16_config = nvfp4a16_config

    @classmethod
    def from_config(cls, config: Dict[str, Any]) -> ModelOptMixedPrecisionConfig:
        # ... 省略前面的解析逻辑 ...
        # 创建 MXFP8 配置，使用 block_size [1, 32] 和 dynamic activation
        mxfp8_config = Fp8Config(
            is_checkpoint_fp8_serialized=True,
            activation_scheme="dynamic",
            weight_block_size=[1, 32],
            packed_modules_mapping=packed_modules_mapping,
            use_mxfp8=True, # <-- 启用 MXFP8 模式
        )
        return cls(
            kv_cache_quant_algo=kv_cache_quant_algo,
            exclude_modules=exclude_modules,
            packed_modules_mapping=packed_modules_mapping,
            quantized_layers=quantized_layers,
            fp8_config=fp8_config,
            mxfp8_config=mxfp8_config, # <-- 传递新配置
            nvfp4_config=nvfp4_config,
            nvfp4a16_config=nvfp4a16_config,
        )

    def get_quant_method(self, layer, prefix):
        quant_algo = self._resolve_quant_algo(prefix)
        # ... 前面的 quant_algo 检查 ...
        if isinstance(layer, LinearBase):
            if quant_algo == "MXFP8":
                return Fp8LinearMethod(self.mxfp8_config) # <-- 新增 MXFP8 线性层方法
        if isinstance(layer, FusedMoE):
            if quant_algo == "MXFP8":
                return Fp8MoEMethod(self.mxfp8_config) # <-- 新增 MXFP8 MoE 层方法
```

python/sglang/srt/models/minimax_m3.py

添加 hf_to_sglang_mapper 解决前缀映射，并禁用混合精度下的共享专家融合

```
from sglang.srt.models.utils import WeightsMapper

class MiniMaxM3SparseForCausalLM(nn.Module):
    # 将 HF 的 block_sparse_moe 前缀映射为 SGLang 的 mlp 前缀，确保量化配置能匹配路由专家层
    hf_to_sglang_mapper = WeightsMapper(
        orig_to_new_substr={".block_sparse_moe.": ".mlp."}
    )
    packed_modules_mapping = {
        "qkv_proj": ["q_proj", "k_proj", "v_proj"],
        "index_qkv_proj": ["index_q_proj", "index_k_proj", "index_v_proj"],
        "gate_up_proj": ["gate_proj", "up_proj"],
    }

    # ... 其他方法 ...

    def determine_num_fused_shared_experts(self):
        if get_server_args().disable_shared_experts_fusion:
            return
        disable_reason = None
        if not getattr(self.config, "n_shared_experts", None):
            disable_reason = "No shared experts are defined in the config."
        # 当使用 ModelOpt 混合精度时，共享专家与路由专家可能采用不同量化格式（如 MXFP8 vs
        NVFP4），
        # 此时必须禁用共享专家融合，避免计算错误
        elif (
            self.quant_config is not None
            and self.quant_config.get_name() == "modelopt_mixed"
        ):
            disable_reason = (
                "Shared and routed experts may use different quantization formats "
                "in ModelOpt mixed-precision checkpoints."
            )
        # ... 其他条件 ...
```

评论区精华

此 PR 无代码评审评论，只有一次审批（b8zhong 批准）。主要讨论在关联 Issue #31827 中，该 Issue 详细描述了问题的两个根因：① fused MoE 量化方法无法解析（因 HF 前缀 block_sparse_moe 未映射到 SGLang 的 mlp）；② MXFP8 线性层在混合精度配置中被忽略。

- 暂无高价值评论线程

风险与影响

- 风险：更改了 modelopt_quant.py 中的配置解析，可能影响其他使用 ModelOptMixedPrecisionConfig 的模型（如 Nemotron-H），但由于添加的是新选项（MXFP8）并保持向后兼容，风险较低。修改了 overrides.py 中 MiniMax-M3 的默认 MoE

后端，但条件严格限定为 modelopt_mixed，不会影响其他量化方法。禁用共享专家融合可能影响性能，但这是混合精度所必需的。

- 影响：用户：能够成功加载并使用 nvidia/MiniMax-M3-NVFP4 模型，精度测试通过（GSM8K 0.925）。系统：SGLang 现在支持包含 MXFP8 层的 modelopt_mixed 混合精度模型，扩展了对更多 NVIDIA 量化模型的支持。团队：新增了少量维护点，但添加了单元测试覆盖，降低了回归风险。
- 风险标记：量化配置变更，MoE 后端默认值变更，共享专家融合禁用

关联脉络

- PR #32036 [AMD] Minimax-M3：unblock mxfp8 block convert on gfx950: 同样涉及 MXFP8 支持，为本 PR 的 MXFP8 配置互动作铺垫
- PR #32230 [AMD] MiniMax-M3: opt-in custom/quick all-reduce on ROCm: 同为 MiniMax-M3 的优化 PR，共享硬件适配上下文