PR #27375 完整报告

sgl-project/sglang

[Model] Add support for JetBrains' Mellum v2 code generation model

合并时间：2026-07-14 13:54

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/27375

执行摘要

- 一句话：新增 JetBrains Mellum v2 代码生成模型推理支持
- 推荐动作：值得关注本 PR 如何通过继承和配置别名降低新模型集成成本，对于类似基于 QwenMoE 架构的模型添加有参考价值。同时 review 中的讨论展示了代码兼容性和正确性审查的必要性。建议后续补充更多端到端测试并考虑 NPU 支持。

功能与动机

添加对 JetBrains/Mellum2-12B-A2.5B-Thinking 及相关变体的支持，该模型采用混合滑动窗口注意力和 MoE 架构，专为代码生成任务设计。PR 从 HuggingFace 加载模型并利用 SGLang 推理框架提供服务。

实现拆解

1. 模型核心实现(python/sglang/srt/models/mellum.py): 新建 MellumForCausalLM 类，继承自 Qwen3MoeForCausalLM。子类 MellumAttention 重写 __init__和 forward_prepare_npu，注入逐层滑动窗口和 per-layer-type RoPE；MellumMLP 修复 forward 签名兼容性。
2. 配置兼容层(python/sglang/srt/utils/hf_transformers/common.py): 在 _CONFIG_REGISTRY 中注册 'mellum' 键。优先使用 transformers>=5.10.2 的原生 MellumConfig，否则创建基于 Qwen3MoeConfig 的别名并重写 __post_init__以确保 sliding_window 不会被错误清零。
3. 模型配置路由(python/sglang/srt/configs/model_config.py): 将 MellumForCausalLM 加入混合 SWA 模型集合并统一化 get_hybrid_layer_ids 的分支逻辑，消除了之前为 LagunaForCausalLM 单独编写的冗余代码。
4. MoE Bench 集成(benchmark/kernels/fused_moe_triton/common_utils.py): 在模型专家数查询表中添加 MellumForCausalLM，使其能正确参与 MoE 性能基准测试。
5. 测试与文档(test/registered/unit/configs/test_model_config.py, test/registered/unit/models/test_mellum.py, docs_new/docs/supported-models/generative_models.mdx): 添加针对混合层 ID 解析和位置准备的单元测试；文档中新增 Mellum 模型支持行。

关键文件：

- python/sglang/srt/models/mellum.py（模块 模型定义；类别 source；类型 data-contract；符号 _get_rope_type, _compute_yarn_from_rope_params, get_attention_sliding_window_size, MellumMLP）：模型核心实现，新增 MellumForCausalLM 等类，定义混合注意力逻辑。
- python/sglang/srt/utils/hf_transformers/common.py（模块 配置兼容；类别 source；类型 dependency-wiring；符号 _MellumConfigAlias, post_init）：配置兼容层，处理 MellumConfig 注册，确保旧版 transformers 下也能加载模型。
- python/sglang/srt/configs/model_config.py（模块 配置路由；类别 source；类型 data-contract）：模型配置路由，将 Mellum 集成到混合 SWA 模型判断中，并进行代码简化。
- test/registered/unit/configs/test_model_config.py（模块 配置测试；类别 test；类型 test-coverage；符号 TestHybridLayerIds, test_layer_type_architectures）：测试混合注意力模型配置层 ID 解析，覆盖 Mellum 在内的多种架构。
- test/registered/unit/models/test_mellum.py（模块 模型测试；类别 test；类型 test-coverage；符号 TestMellumForCausalLM, test_prepare_positions）：Mellum 模型单元测试，验证 position 准备逻辑。
- benchmark/kernels/fused_moe_triton/common_utils.py（模块 基准配置；类别 source；类型 core-logic）：MoE 基准测试配置，添加 Mellum 架构以识别专家数。
- docs_new/docs/supported-models/generative_models.mdx（模块 文档；类别 other；类型 data-contract）：文档更新，列出 Mellum 为支持模型。

关键符号：MellumMLP.forward, MellumAttention.init, MellumAttention.forward_prepare_npu, _get_rope_type, _compute_yarn_from_rope_params, get_attention_sliding_window_size, _MellumConfigAlias.post_init, get_hybrid_layer_ids

关键源码片段

python/sglang/srt/models/mellum.py

模型核心实现，新增 MellumForCausalLM 等类，定义混合注意力逻辑。

```
# MellumMLP: 简单的 forward 签名适配层
# Qwen3MoeDecoderLayer.forward 调用 self.mlp(x, forward_batch),
# 但 Qwen3MoeMLP.forward 只接受 x。
# MellumMLP 桥接这一差异以兼容父类 DecoderLayer。
class MellumMLP(Qwen3MoeMLP):
    def forward(self, x, forward_batch=None):
        return super().forward(x)


def _get_rope_type(rope_params: Dict[str, Any]) -> str:
    '''提取 rope_type，优先使用 'rope_type' 键，回退到 'type' 或默认值。'''
    return rope_params.get('rope_type') or rope_params.get('type') or 'default'
```

python/sglang/srt/utils/hf_transformers/common.py

配置兼容层，处理 MellumConfig 注册，确保旧版 transformers 下也能加载模型。

```
# 当 transformers 版本过低时，使用基于 Qwen3MoeConfig 的别名类
# 并保留 sliding_window 属性以免被父类 __post_init__ 清除
if _HFMellumConfig is not None:
    _CONFIG_REGISTRY['mellum'] = _HFMellumConfig
else:
    from transformers import Qwen3MoeConfig as _HFQwen3MoeConfig

    class _MellumConfigAlias(_HFQwen3MoeConfig):
        model_type = 'mellum'

        def __post_init__(self, **kwargs):
            # Qwen3MoeConfig.__post_init__ 会清除 sliding_window，
            # 除非 use_sliding_window=True。Mellum 通过 layer_types
            # 逐层控制滑动注意力，因此必须保留 sliding_window。
            sliding_window = getattr(self, 'sliding_window', None)
            super().__post_init__(**kwargs)
            self.sliding_window = sliding_window

    _CONFIG_REGISTRY['mellum'] = _MellumConfigAlias
```

评论区精华

Review 中主要讨论集中在三个问题：

- gate_up_proj 应定义在类级 packed_modules_mapping 而非实例级，以避免量化框架和权重加载器出错。最终修复。
- _MellumConfigAlias.__post_init__ 签名兼容性：初始版本使用 **kwargs 引发 TypeError，尝试修复后又导致 head_dim 回归，最终版本通过调整参数避免问题。
- 代码简化和位置提升建议：get_hybrid_layer_ids 分支合并已完成，位置提升作为 TODO 保留。
- gate_up_proj 定义位置 (design): Jiminator 在后续提交中将 gate_up_proj 移至类级别 packed_modules_mapping，问题解决。
- __post_init__签名兼容 (correctness): 最终版本保留了 __post_init__但调整了参数调用方式，避免了 **kwargs 导致的错误。
- 代码简化与位置提升 (design): 分支合并已完成（统一使用 layer_types 逻辑），位置提前作为 TODO 保留。

风险与影响

- 风险：配置兼容风险：旧版 transformers 回退路径中的 __post_init__ 签名问题曾引发回归，虽已修正但需谨慎验证。NPU 路径被显式拒绝，在 NPU 设备上直接使用会出错。单元测试仅覆盖位置预备和混合层 ID 解析，未包含注意力计算和 MLP 端到端正确性测试。新模型注册可能被未知 transformers 版本打断。

- 影响：对用户：支持使用 Mellum v2 模型进行推理，调用方式与 Qwen3MoE 模型一致。对系统：新增模型配置注册影响可忽略。对团队：维护了代码一致性，但新增模型特定代码需要持续维护。对已有功能：完全向后兼容。
- 风险标记：配置兼容风险，NPU 未测试，测试覆盖有限，依赖旧版 transformers 回退

关联脉络

- 暂无明显关联 PR