PR #25061 完整报告

sgl-project/sglang

Fix MiniMax-M2.7 on CPU

合并时间：2026-05-28 10:53

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/25061

执行摘要

- 一句话：修复 MiniMax-M2.7 在 CPU 上的运行问题
- 推荐动作：该 PR 值得合并，它解决了特定模型的 CPU 兼容性，且改动集中、设计合理。建议关注作者创建的 issue #26439 以跟踪后续优化（如融合 all-reduce、增加 CPU kernel 支持）。Review 中关于类型转换隐藏和条件精度的做法值得借鉴。

功能与动机

MiniMax-M2.7 模型在 CPU 设备上无法正常运行，因为现有的 CPU kernel 不支持某些 top-k 配置（correction_bias 非 None 或 scoring_func 不是 softmax），且模型的部分算子（如 MiniMaxM2QKRMSNorm）缺乏 CPU 实现。PR 旨在修复这些问题，使模型能在 CPU 上推理。

实现拆解

1. top-k 路由 fallback（python/sglang/srt/layers/moe/topk.py）：在 fused_topk_cpu 函数中添加条件判断，当 correction_bias is not None 或 scoring_func != "softmax" 时回退到 fused_topk_torch_native，因为 CPU kernel 尚不支持这些特性。MiniMax 使用了自定义路由函数，由此获得正确结果。
2. CPU 专用前向路径（python/sglang/srt/models/minimax_m2.py）：为 MiniMaxM2QKRMSNorm 新增 _forward_cpu 方法，直接调用普通的 QK 归一化（.contiguous() 后依次执行 _q_norm 和 _k_norm），避免依赖 Triton 融合 kernel。通过 _is_cpu 和 _is_amx_available 标志在 __init__ 中分发到该方法。
3. 权重加载适配（python/sglang/srt/models/minimax_m2.py）：在 MiniMaxM2RMSNormTP.weight_loader 中，当 CPU 且 AMX 可用时，使用 narrow_padded_param_and_loaded_weight 处理不均匀 TP 分片，确保权重正确加载。
4. C++ kernel 类型兼容（sgl-kernel/csrc/cpu/moe.cpp）：在 fused_experts_cpu 入口处增加 topk_ids_ 转换：若 topk_ids 为 int64（来自 torch native fallback），自动转换为 int32，满足 kernel 内部要求。这样隐藏了类型转换细节，保持 Python 接口整洁。
5. 环境检测支持（python/sglang/srt/models/minimax_m2.py）：新增 cpu_has_amx_support 和 is_cpu 模块级变量，用于后续条件判断。

关键文件：

- python/sglang/srt/models/minimax_m2.py（模块 模型层；类别 source；类型 core-logic；符号 _forward_cpu）：核心模型文件，新增 CPU 前向路径和权重加载适配，是本次改动

的入口。

- python/sglang/srt/layers/moe/topk.py（模块 MoE 路由；类别 source；类型 core-logic）：Top-k 路由 fallback，核心修改使 MiniMax 的自定义路由函数能在 CPU 上正确降级到 torch native。
- sgl-kernel/csrc/cpu/moe.cpp（模块 CPU 内核；类别 source；类型 core-logic）：C++ kernel 层增加 int64 兼容，使 torch native fallback 产生的 topk_ids 能无缝接入 fused_experts_cpu。

关键符号：_forward_cpu, weight_loader, fused_topk_cpu, fused_experts_cpu

关键源码片段

python/sglang/srt/models/minimax_m2.py

核心模型文件，新增 CPU 前向路径和权重加载适配，是本次改动的入口。

```
# python/sglang/srt/models/minimax_m2.py
# _forward_cpu: CPU 专用的 QK 归一化，不使用 Triton 融合 kernel
# 当 _is_cpu 且 _is_amx_available 时，在 __init__ 中分配到该方法
# TODO: 融合 q 和 k 的 all-reduce 以减少通信开销
def _forward_cpu(self, q: torch.Tensor, k: torch.Tensor):
    # 先确保内存连续，再依次调用归一化
    q = self._q_norm(q.contiguous())
    k = self._k_norm(k.contiguous())
    return q, k

# weight_loader 中的 CPU 分支：利用 narrow_padded_param_and_loaded_weight 处理不均匀 TP 分片
def weight_loader(
    self,
    param: nn.Parameter,
    loaded_weight: torch.Tensor,
) -> None:
    shard_id = self.attn_tp_rank // self.num_head_replicas
    shard_size = param.data.shape[0]

    if _is_cpu and _is_amx_available:
        # 仅在 CPU + AMX 时需要特殊处理
        param_data, loaded_weight = narrow_padded_param_and_loaded_weight(
            param.data,
            loaded_weight,
            0, # param_data_start
            shard_id * shard_size, # weight_start
            0, # shard_axis
            shard_size,
        )
        param_data.copy_(loaded_weight)
        return
    # 其余路径保持不变（GPU/ 普通 CPU）
```

```
    shard_end = (shard_id + 1) * shard_size
    assert shard_end <= loaded_weight.shape[0], ...
    param.data.copy_(loaded_weight[shard_id * shard_size:shard_end])
```

python/sglang/srt/layers/moe/topk.py

Top-k 路由 fallback，核心修改使 MiniMax 的自定义路由函数能在 CPU 上正确降级到 torch native。

```
# python/sglang/srt/layers/moe/topk.py -- fused_topk_cpu 函数
# 新加的条件：若 kernel 不支持 correction_bias 或非 softmax 评分函数，则回退到 torch native
def fused_topk_cpu(
    hidden_states: torch.Tensor,
    gating_output: torch.Tensor,
    topk: int,
    renormalize: bool,
    correction_bias: torch.Tensor = None,
    scoring_func: str = "softmax",
):
    # TODO: add c++ kernel for cpu
    # The topk_softmax_cpu kernel only handles vanilla softmax scoring with no
    # correction bias. Fall back to the torch-native impl for the rest
    # (e.g. MiniMax sets both correction_bias and scoring_func).
    if correction_bias is not None or scoring_func != "softmax":
        return fused_topk_torch_native(
            hidden_states,
            gating_output,
            topk,
            renormalize,
            correction_bias=correction_bias,
            scoring_func=scoring_func,
        )
    # 标准情况：使用优化的 CPU kernel
    topk_weights, topk_ids = torch.ops.sgl_kernel.topk_softmax_cpu(
        hidden_states=hidden_states,
        gating_output=gating_output,
        topk=topk,
        renormalize=renormalize,
    )
    return topk_weights, topk_ids
```

sgl-kernel/csrc/cpu/moe.cpp

C++ kernel 层增加 int64 兼容，使 torch native fallback 产生的 topk_ids 能无缝接入 fused_experts_cpu。

```
// sgl-kernel/csrc/cpu/moe.cpp -- fused_experts_cpu 函数入口
// 自动处理 topk_ids 类型：若 torch native fallback 产生 int64，则转为 int32
// TODO: Remove the typecast after topk kernel is provided
const auto st = hidden_states.scalar_type();
auto topk_ids_ = topk_ids.scalar_type() == at::kInt ? topk_ids : topk_ids.to(at::kInt);
```

```
CHECK_INPUT(hidden_states);
CHECK_INPUT(w1);
CHECK_INPUT(w2);
CHECK_EQ(topk_weights.sizes(), topk_ids_.sizes());
// ... 后续所有使用 topk_ids 的地方都替换为 topk_ids_
```

评论区精华

Review 中主要讨论了以下要点：

- int64 转换位置：mingfeima 建议将 topk_ids 的 int64 转换隐藏在 fused_experts_cpu 内部（C++ 层），而不是暴露在 Python 的 fp8.py 中，以保持高层代码清洁。作者采纳并移入 moe.cpp。
- 条件检查精度：mingfeima 建议将 weight_loader 中的 if _is_cpu: 改为 if _is_cpu and _is_amx_available:，因为仅当 AMX 可用时才需要特殊处理。作者已修改。
- pin_memory 可移植性：mingfeima 建议使用平台接口 is_pin_memory_available() 替代 not _is_cpu。作者 rebase 后发现主线已修复该问题，故未额外改动。
- 融合 all-reduce 优化：gemini-code-assist 建议在 _forward_cpu 中融合 q 和 k 的 all-reduce 以减少通信开销。作者添加了 TODO 注释，留作后续优化。
- int64 转换位置：从 Python 移至 C++ (design): 作者采纳，将转换逻辑移入 moe.cpp。
- CPU 条件检查应同时判断 AMX 支持 (correctness): 作者采纳并修改。
- 使用 is_pin_memory_available 接口替代 not _is_cpu (design): 作者 rebase 后发现主线已修复该问题，故未单独修改。
- _forward_cpu 中融合 all-reduce 优化 (performance): 作者添加 TODO 注释，暂未实现，留后续优化。

风险与影响

- 风险：
  - 临时回退路径：fused_topk_cpu 中的 torch native fallback 在非标准 top-k 配置下性能较低，但确保了正确性；后续需为 CPU 提供 kernel 支持以消除回退。
  - 类型转换复杂性：moe.cpp 中增加的 int64 转 int32 逻辑仅影响 fallback 路径，但引入了 kernel 入口的冗余判断；若后续统一 all kernel 输入类型可简化。
  - CPU 前向性能：_forward_cpu 目前使用普通 PyTorch 操作，没有融合 kernel，可能在分布式场景下因多次 all-reduce 导致额外通信开销。
  - 缺少测试覆盖：本次改动未包含测试文件，CPU 路径的正确性依赖手动验证，存在回归风险。
- 影响：
  - 用户影响：MiniMax-M2.7 模型现在可在 CPU 设备上推理，扩展了可部署硬件范围；但性能可能不如 GPU。
  - 系统影响：改动局限在模型文件和 kernel 层，未影响核心调度或缓存逻辑。
  - 团队影响：引入了临时的 fallback 模式，需注意后续 kernel 升级时清理这些回退路径。

- 风险标记：缺少测试覆盖，临时回退路径，CPU fallback 性能风险

关联脉络

- 暂无明显关联 PR