PR #19484 完整报告

sgl-project/sglang

[CPU] Add Qwen3.5 model optimization for CPU

合并时间：2026-04-27 01:12

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/19484

执行摘要

- 一句话：CPU 优化 Qwen3.5 系列：TP padding 和新 kernel
- 推荐动作：值得精读，尤其是 resolve_head_dim 的提取、pad_loaded_weight 的谨慎设计、以及 TP padding 在 CPU 下的交互。Review 中涵盖的多模态配置边界条件和 assert 陷阱对所有贡献者都有参考价值。

功能与动机

PR body 明确指出需要为 Qwen3.5 系列添加 CPU 优化的支持，包括 dtype 支持、融合 kernel、TP padding 和日志改进。评论中提到这是 Intel CPU 上 Qwen3.5 系列启动的最后一个 PR。

实现拆解

1. 新增 CPU kernel：在 sgl-kernel/csrc/cpu/model/qwen3.cpp 中添加 fused_qkvzba_split_reshape_cat_contiguous_cpu，用于连续布局的 QKV+BA 拆分，同时添加 dtype 一致性检查；
2. 数据类型扩展：在 sgl-kernel/csrc/cpu/mamba/fla.cpp 中将 fused_sigmoid_gating_delta_rule_update 和 fused_gdn_gating 泛化以支持 bf16 A_log，并新增标量回退分支；
3. TP padding 机制：在 update_config.py 中重构 get_num_heads_padding_size，新增 resolve_head_dim 辅助函数；在 weight_utils.py 中新增 pad_loaded_weight 函数，并在 sharded_weight_loader 中扩展 padding 条件。这些更改使 CPU 在不均衡 TP 场景下能正确 padding；
4. 模型层适配：修改 qwen3_vl.py 和 qwen3_5.py 等模型，注入 CPU 特定逻辑（AMX 检测、padded_context_dim 传递、tie_word_embeddings 条件调整）；
5. 工具与日志：在 common.py 中添加 log_debug_on_rank0，并改进 log_info_on_rank0 的异常处理；
6. 测试配套：在 test/srt/cpu/test_qwen3.py 中新增 test_fused_qkvzba_split_reshape_cat_contiguous 和辅助函数。

关键文件：

- python/sglang/srt/configs/update_config.py（模块 配置层；类别 source；类型 core-logic；符号 get_num_heads_padding_size, resolve_head_dim, update_config）：

核心配置逻辑变更，重构了 head_dim 解析和 padding 大小计算，新增resolve_head_dim 辅助函数

- python/sglang/srt/model_loader/weight_utils.py（模块 权重加载；类别 source；类型 data-contract；符号 pad_loaded_weight）：新增 pad_loaded_weight 函数用于 TP padding 时的权重补齐，并扩展 sharded_weight_loader 的 padding 条件
- sgl-kernel/csrc/cpu/model/qwen3.cpp（模块 CPU Kernel；类别 source；类型 core-logic；符号 fused_qkvzba_split_reshape_cat_contiguous_cpu）：新增 fused_qkvzba_split_reshape_cat_contiguous_cpu 融合 kernel，是性能优化的核心
- python/sglang/srt/utils/common.py（模块 工具函数；类别 source；类型 dependency-wiring；符号 log_debug_on_rank0）：新增 log_debug_on_rank0 工具函数，改进 log_info_on_rank0 的异常处理
- sgl-kernel/csrc/cpu/mamba/fla.cpp（模块 CPU Kernel；类别 source；类型 core-logic）：泛化 fused_sigmoid_gating_delta_rule_update 和 fused_gdn_gating 支持 bf16 A_log，并新增标量回退 kernel
- python/sglang/srt/models/qwen3_vl.py（模块 视觉模型；类别 source；类型 data-contract）：添加 CPU/AMX 环境检测，传递 head_size 和 padded_context_dim，调整 tie_word_embeddings 条件
- python/sglang/srt/models/qwen3_5.py（模块 模型定义；类别 source；类型 data-contract）：调整 TP padding 分流逻辑和 tied embedding 权重拷贝条件
- test/srt/cpu/test_qwen3.py（模块 测试；类别 test；类型 test-coverage；符号 fix_query_key_value_ordering_reshape_cat_contiguous, test_fused_qkvzba_split_reshape_cat_contiguous）：新增 test_fused_qkvzba_split_reshape_cat_contiguous 测试用例，验证新 kernel 的正确性

关键符号：get_num_heads_padding_size, resolve_head_dim, update_config, pad_loaded_weight, fused_qkvzba_split_reshape_cat_contiguous_cpu, fused_sigmoid_gating_delta_rule_update, fused_gdn_gating_kernel, log_debug_on_rank0

关键源码片段

python/sglang/srt/configs/update_config.py

核心配置逻辑变更，重构了 head_dim 解析和 padding 大小计算，新增 resolve_head_dim 辅助函数

```
def resolve_head_dim(cfg, num_heads, is_text_config):
    # 默认通过 hidden_size 和 num_heads 计算 head_dim，兼容 d_model 属性
    hidden_size = getattr(cfg, "hidden_size", getattr(cfg, "d_model", None))
    head_dim = hidden_size // num_heads if hidden_size else None
    # 文本配置优先使用 qk_head_dim 或 head_dim 属性
    if is_text_config:
        if hasattr(cfg.hf_config, "qk_head_dim"):
            head_dim = cfg.hf_config.qk_head_dim
        elif hasattr(cfg.hf_text_config, "head_dim"):
            head_dim = cfg.hf_text_config.head_dim
        elif hasattr(cfg.hf_config, "head_dim"):
```

```
            head_dim = cfg.hf_config.head_dim
    else:
        # 非文本配置（vision/audio）直接使用 head_dim 属性
        if hasattr(cfg, "head_dim"):
            head_dim = cfg.head_dim
    return head_dim


def get_num_heads_padding_size(tp_size, weight_block_size, head_dim=None):
    # 当 head_dim 为 None 时（来自非文本配置），使用简单规则
    if head_dim is None:
        pad_size = (
            tp_size * 2
            if tp_size % 2 == 1 and weight_block_size is not None
            else tp_size
        )
        return pad_size
    pad_size = tp_size
    # 考虑 weight_block_size 对齐要求，用 lcm 扩展 pad_size
    if weight_block_size is not None and head_dim % weight_block_size[0] != 0:
        import math
        pad_size = tp_size * (
            math.lcm(head_dim, weight_block_size[0]) // weight_block_size[0]
        )
    return pad_size
```

python/sglang/srt/model_loader/weight_utils.py

新增 pad_loaded_weight 函数用于 TP padding 时的权重补齐，并扩展 sharded_weight_loader 的 padding 条件

```
def pad_loaded_weight(loaded_weight, output_dim, output_sizes):
    # 当 loaded_weight 小于 output_sizes 时进行零填充，用于 TP 非对齐场景
    total_output_size = sum(output_sizes)
    raw_output_size = loaded_weight.size(output_dim)
    if total_output_size > raw_output_size:
        # 按比例分割 raw 部分，必须保证总和相等（否则报错）
        weight_split_size = [
            int(output_size / total_output_size * raw_output_size)
            for output_size in output_sizes
        ]
        assert (
            sum(weight_split_size) == raw_output_size
        ), f"Padding failed: sum splits {sum(weight_split_size)} != raw size {raw_output_size}"
        split_weight = loaded_weight.split_with_sizes(weight_split_size, dim=output_dim)
        padded_parts = []
        for i, output_size in enumerate(output_sizes):
            pad_size = output_size - weight_split_size[i]
            if pad_size > 0:
                pad_shape = list(loaded_weight.size())
```

```
                pad_shape[output_dim] = pad_size
                pad_tensor = torch.zeros(pad_shape, dtype=loaded_weight.dtype)
                padded_parts.append(torch.cat([split_weight[i], pad_tensor], dim=output_dim))
            else:
                padded_parts.append(split_weight[i])
        return torch.cat(padded_parts, dim=output_dim)
    else:
        return loaded_weight
```

评论区精华

Review 中聚焦在多个正确性问题上：JustinTong 指出 head_dim 解析在多模态子配置下可能 UnboundLocalError（update_config.py），建议兼容 d_model，作者接受并提取为 resolve_head_dim；pad_loaded_weight 的比例分割可能导致无声截断，添加 assert 后修复；多处 assert 使用了永真元组 (cond, msg,) 导致不触发，已全部修正；CPU-without-AMX 的 tie_word_embeddings 条件缺陷导致 lm_head 可能未初始化，改为对称条件；日志在循环内重复且级别不当，已提升循环外并改为 warning；异常处理从 bare except 改为 except Exception 以兼顾 RuntimeError 和 AssertionError。还有测试未注册 CPU CI 的问题被推迟到后续 PR。

- head_dim 解析兼容多模态配置 (correctness): 已采纳并提取为 resolve_head_dim 函数
- pad_loaded_weight 分割舍入错误 (correctness): 添加了 assert 并修正了 qwen3_5.py 和 mamba.py 中的类似代码
- assert 格式错误 (correctness): 已修正所有出现点
- CPU-without-AMX tie_word_embeddings 条件 (correctness): 改为 and not (_is_cpu and _is_cpu_amx_available)，并补全权重拷贝
- 日志重复与级别 (performance): 已修正：日志移出循环，级别改为 warning
- 异常处理范围 (correctness): 改为 except Exception 并优化消息格式
- 测试未注册到 CPU CI (testing): 作者表示将在后续 PR 中跟进（WIP refactoring）

风险与影响

- 风险：pad_loaded_weight 的整数分割即使有 assert 在极端比例下也可能触发异常导致加载失败；resolve_head_dim 若返回 None 可能导致后续 padding 逻辑异常；新 kernel 的测试缺少多 TP 尺寸覆盖；log_debug_on_rank0 的异常处理修改可能暴露不期望的异常；sharded_weight_loader 中 padding 条件扩展只在 CPU 激活，可能被误解为 CPU 通用代码。
- 影响：对 CPU 用户可以运行 Qwen3.5 系列（含视觉模型）并获得 TP 非对齐支持；对系统增加了 SGL-Kernel 和权重加载的 CPU 专用路径；对团队提高了配置更新与日志可维护性，但增加了多条件分支的维护负担。
- 风险标记：split 舍入 assert 可能失败，未测试多 TP 尺寸，多模态 head_dim 缺省值，assert 格式历史风险

关联脉络

- PR #12531 [CPU] mrope kernel support: 本 PR 依赖该 PR，先决条件提供 CPU mrope kernel
- PR #19070 fix accuracy issue on main: Review 中提及该 PR 已修复准确性问题，删除了部分 workaround