PR #12662 完整报告

sgl-project/sglang

[CPU] Add support for Qwen3-vl and Qwen3-omni

合并时间：2026-05-27 08:56

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/12662

执行摘要

- 一句话：为 CPU 添加 Qwen3-VL / Omni 前端支持
- 推荐动作：值得详细阅读。代码设计体现了对异构硬件后端（CPU AMX）的良好抽象（通过 QKV_BACKEND_IMPL 字典），并且在 Review 中展现了在性能与代码干净度之间权衡的典型过程（移除 hack、defer 性能问题）。为后续多模态模型 CPU 支持提供了模板。

功能与动机

在 Intel CPU 上运行 Qwen3-VL 和 Qwen3-Omni 多模态模型。此前已有多个 CPU AMX 优化基础 PR（如 #15168、#17919、#16040 等），需要将这些优化集成到模型前端的视觉和音频编码器中，填补多模态模型在 CPU 上的推理空白。

实现拆解

1. 集成 CPU AMX 注意力后端：在 python/sglang/srt/layers/attention/vision.py 中新增 VisionAMXAttention 类，在 CPU AMX 环境下通过 sgl-kernel 的 flash_attn_varlen_func 实现视觉注意力的高性能计算，并将该后端注册到全局 QKV_BACKEND_IMPL 分发字典中。
2. 修复 Tensor Parallelism 的填充问题：在 python/sglang/srt/models/qwen3_omni_moe.py 和 qwen3_vl.py 中，引入 get_head_dim_and_projection_size 函数。当 tp_size=3/6 导致 num_heads 被填充时，保持原始 head_dim 并重新计算 projection_size，避免张量形状与权重检查点不兼容。
3. 替换 Linear 层以兼容 AMX 权重打包：在 qwen3_omni_moe.py 中，将音频编码器 FFN 层从 ColumnParallelLinear/RowParallelLinear 有条件地替换为 ReplicatedLinear，以利用 AMX 后端的 weight_packed_linear 路径。同时替换 conv_out 和 proj1 等层。
4. 植入 CPU 加速的图像预处理：在 python/sglang/srt/layers/amx_utils.py 中封装 fast_preprocess_cpu 函数，并在 qwen_vl.py 中通过 monkey-patch 替换 Qwen2VLImageProcessorFast._preprocess，对 CPU AMX 场景透明生效。
5. 优化 Conv3d Patch Embedding：在 python/sglang/srt/layers/conv.py 中增加 forward_cpu 路径，调用 sgl-kernel 的 conv3d_embed_cpu 算子，并补充 5 维卷积权重的 packing 逻辑。

关键文件：

- python/sglang/srt/layers/attention/vision.py（模块 注意力层；类别 source；类型 core-logic；符号 VisionAMXAttention, init, forward）：核心变更模块。新增 VisionAMXAttention 类，注册到 QKV_BACKEND_IMPL 分发字典；导入 cpu_has_amx_support、is_cpu；对 CPU AMX 场景选择 amx_attn 后端。同时清理了 reshape 后的 contiguous() 调用。
- python/sglang/srt/models/qwen3_omni_moe.py（模块 模型定义；类别 source；类型 data-contract；符号 get_head_dim_and_projection_size, init）：模型定义关键调整。引入 get_head_dim_and_projection_size 工具函数解决 TP 填充导致的形状不匹配问题。将 nn.Linear / ColumnParallelLinear 替换为 ReplicatedLinear 以启用 AMX 权重 packing 路径。调整 VisionAttention 构造参数。
- python/sglang/srt/layers/amx_utils.py（模块 工具函数；类别 source；类型 core-logic；符号 fast_preprocess_cpu）：新增 CPU 图像预处理关键算子 fast_preprocess_cpu 的 Python 封装。增强 amx_process_weight_after_loading 对 5 维 conv3d 权重的支持。调整 is_dim_conv_weight 函数。
- python/sglang/srt/layers/conv.py（模块 卷积层；类别 source；类型 core-logic；符号 forward_cpu）：为 Conv3d patch embedding 添加 CPU forward 路径 forward_cpu，在 AMX 可用时调用 sgl-kernel 的 conv3d_embed_cpu，否则回退原生逻辑。
- python/sglang/srt/multimodal/processors/qwen_vl.py（模块 预处理；类别 source；类型 dependency-wiring）：将图像预处理加速通过 monkey-patch 注入 Qwen2VLImageProcessorFast，实现 CPU AMX 场景的透明加速。
- python/sglang/srt/models/qwen3_vl.py（模块 模型定义；类别 source；类型 data-contract）：为 Qwen3-VL 的 vision encoder 适配 TP 填充问题，并根据 CPU AMX 环境选择 LayerNorm 实现。
- python/sglang/srt/layers/logits_processor.py（模块 推理层；类别 source；类型 dependency-wiring）：修正了基于 is_cpu 的设备检查逻辑，确保 logits 处理在 CPU 上不走 CUDA 路径。
- python/sglang/srt/server_args.py（模块 服务配置；类别 source；类型 core-logic）：调整了与 CPU 启动相关的参数设置，属于配套修改。

关键符号：VisionAMXAttention.init, VisionAMXAttention.forward, fast_preprocess_cpu, forward_cpu (in conv.py), get_head_dim_and_projection_size

关键源码片段

python/sglang/srt/layers/attention/vision.py

核心变更模块。新增 VisionAMXAttention 类，注册到 QKV_BACKEND_IMPL 分发字典；导入 cpu_has_amx_support、is_cpu；对 CPU AMX 场景选择 amx_attn 后端。同时清理了 reshape 后的 contiguous() 调用。

```
# python/sglang/srt/layers/attention/vision.py
# VisionAMXAttention class & backend registration

import torch
import torch.nn as nn
```

```
from sglang.srt.utils import cpu_has_amx_support, is_cpu

_is_cpu = is_cpu()
_is_cpu_amx_available = cpu_has_amx_support()

# If running on CPU with AMX, use sgl-kernel's flash attention
if _is_cpu and _is_cpu_amx_available:
    flash_attn_varlen_func = torch.ops.sgl_kernel.flash_attn_varlen_func


class VisionAMXAttention(nn.Module):
    """CPU AMX-based attention backend for vision transformers.

    Only available on Intel CPUs with AMX support (e.g., Sapphire Rapids).
    Delegates to `sgl-kernel`'s `flash_attn_varlen_func` for efficient
    variable-length multi-head attention computation.
    """

    def __init__(self, **kwargs):
        # Guard: prevent instantiation on non-CPU/AMX hardware
        if not _is_cpu or not _is_cpu_amx_available:
            raise Exception(
                "VisionAMXAttention is only available for cpu with amx support"
            )
        super().__init__()

    def forward(
        self,
        q: torch.Tensor,
        k: torch.Tensor,
        v: torch.Tensor,
        cu_seqlens: torch.Tensor | SingletonCache | None,
        bsz: int,
        seq_len: int,
        **kwargs,
    ) -> torch.Tensor:
        # Resolve cu_seqlens (cumulative sequence lengths)
        if cu_seqlens is None:
            cu_seqlens = _get_cu_seqlens_for_shape(bsz, seq_len, device=q.device)
        elif isinstance(cu_seqlens, SingletonCache):
            if cu_seqlens.empty():
                cu_seqlens.set_data(
                    _get_cu_seqlens_for_shape(bsz, seq_len, device=q.device)
                )
            cu_seqlens = cu_seqlens.get_data()

        cu_seqlens = cu_seqlens.to(dtype=torch.int32).to(q.device)
        seq_lens = cu_seqlens[1:] - cu_seqlens[:-1]
        max_seqlen = seq_lens.max().item()
```

```
        # Dispatch to the sgl-kernel CPU AMX flash attention op
        output = flash_attn_varlen_func(
            q,
            k,
            v,
            cu_seqlens_q=cu_seqlens,
            cu_seqlens_k=cu_seqlens,
            max_seqlen_q=max_seqlen,
            max_seqlen_k=max_seqlen,
            causal=False,
        )
        return output


# Register the new backend in the dispatch map
QKV_BACKEND_IMPL = {
    # ... other backends ...
    "amx_attn": VisionAMXAttention, # NEW: CPU AMX backend
}

# In _determine_attention_backend:
# elif _is_cpu and _is_cpu_amx_available:
# backend = "amx_attn"
```

python/sglang/srt/models/qwen3_omni_moe.py

模型定义关键调整。引入 get_head_dim_and_projection_size 工具函数解决 TP 填充导致的形状不匹配问题。将 nn.Linear / ColumnParallelLinear 替换为 ReplicatedLinear 以启用 AMX 权重 packing 路径。调整 VisionAttention 构造参数。

```
# python/sglang/srt/models/qwen3_omni_moe.py
# TP padding compensation and Linear layer replacement

from sglang.srt.distributed import get_tensor_model_parallel_world_size
from sglang.srt.layers.linear import ReplicatedLinear
from sglang.srt.utils import is_cpu

_is_cpu = is_cpu()


def get_head_dim_and_projection_size(
    embed_dim: int,
    num_heads: int,
    original_num_heads: Optional[int] = None,
) -> Tuple[Optional[int], int]:
    if (not _is_cpu) or original_num_heads is None:
        return None, embed_dim

    # On CPU, TP may pad num_heads (e.g. for tp=3/6). In that case we keep the
    # original per-head width (from original_num_heads) and recompute projection_size
```

```
    # with padded num_heads, so attention tensor shapes stay TP-friendly while
    # preserving checkpoint semantics.
    head_dim = embed_dim // original_num_heads
    projection_size = num_heads * head_dim
    return head_dim, projection_size


# Inside Qwen3OmniMoeAudioEncoderLayer.__init__
head_dim, projection_size = get_head_dim_and_projection_size(
    embed_dim=embed_dim,
    num_heads=config.encoder_attention_heads,
    original_num_heads=getattr(config, "original_encoder_attention_heads", None),
)
self.self_attn = VisionAttention(
    embed_dim=embed_dim,
    num_heads=config.encoder_attention_heads,
    head_dim=head_dim,
    projection_size=projection_size,
    use_qkv_parallel=True,
    proj_bias=True,
    flatten_batch=True,
)
# For FFN layers, use ReplicatedLinear if dim cannot be evenly split across TP
tp_size = get_tensor_model_parallel_world_size()
use_replicated = config.encoder_ffn_dim % tp_size != 0
fc1_cls = ReplicatedLinear if use_replicated else ColumnParallelLinear
self.fc1 = fc1_cls(embed_dim, config.encoder_ffn_dim, quant_config=quant_config, bias=True)
```

评论区精华

> VisionAMXAttention 是否重复: mingfeima 询问是否与 jianan-gu 之前的 SDAP 注意力重复。jianan-gu 澄清不重复，本 PR 是新的高性能 AMX 后端。
>
> linear_gelu_linear 融合移除: mingfeima 认为该融合实现 like a hack，要求从当前 PR 剥离。blzheng 在后续 commit 中已回退。
>
> ReplicatedLinear 的性能权衡: mingfeima 指出在 qwen3_omni_moe.py 中使用 ReplicatedLinear 在 tp>1 时可能引入额外通信开销，属于性能退化。blzheng 确认是为了走 AMX weight_packed_linear 路径。mingfeima 标记为 fix this later，当前接受此权衡。

- VisionAMXAttention 是否重复 (design): jianan-gu 澄清不重复，本 PR 是新的高性能 AMX 后端，之前的 SDAP 方案可切换至此
- 移除 linear-gelu-linear fusion (design): blzheng 已在后续 commit c638a0f 中回退此部分
- ReplicatedLinear 的性能权衡 (performance): blzheng 确认这是为了走 AMX 权重打包路径，mingfeima 接受并标记为 'fix this later'
- 图像预处理代码组织 (style): 已采纳，最终从 amx_utils.py 导入函数
- Head size 调整代码提取 (testing/style): 已采纳，新增独立工具函数

风险与影响

- 风险：
  - 回归风险：所有特化代码均被 _is_cpu 和 _is_cpu_amx_available 条件严格守卫，GPU/XPU 等后端不受影响。
  - 性能风险：ReplicatedLinear 替代 ColumnParallelLinear 会导致 FFN 层在 tp>1 时失去 Tensor Parallel 的通信缩减能力，带来性能退化（已认知并 deferred）。
  - 兼容性风险：monkey-patch 直接修改 transformers 库的内部方法 Qwen2VLImageProcessorFast._preprocess，当 transformers 版本更新时可能失效。
  - 测试覆盖：本次变更缺乏对 CPU 端多模态模型的 E2E 集成测试或精度基准测试。
  - 影响：用户侧：Intel CPU 用户可在无 GPU 环境下部署 Qwen3-VL/Omni，且推理速度显著快于纯 PyTorch 实现。

系统侧：强化了对 sgl-kernel 的 CPU 后端依赖。后续模型在 CPU 上实现多模态推理时，可复用本 PR 建立的 AMX 后端注册和算子替换模式。

团队侧：展示了多维度优化（kernel/ 数据通路 / 模型定义）的集成打法和 Review 标准（如移除 hacky 实现、抽象工具函数）。

- 风险标记：非 CPU 场景有隔离守卫，ReplicatedLinear 性能退化已 defer，Monkey-patch 依赖 transformers 内部 API，缺失 E2E 测试覆盖

关联脉络

- PR #15168 optimization for image preprocessor：本 PR 应用了其图像预处理器优化
- PR #17919 Add fused kernel for linear-gelu-linear：本 PR 最初依赖但后被 review 要求移除
- PR #16040 optimization for conv3d：本 PR 应用了其 Conv3d 优化
- PR #13121 remove contiguous overhead before rope：本 PR 应用了其移除 contiguous 的优化
- PR #15075 optimization for layernorm：本 PR 应用了其 LayerNorm 优化
- PR #26132 Sgl flashmla：同属 sgl-kernel 集成到推理路径的典型模式