PR #22498 完整报告

sgl-project/sglang

[CPU] Add support for Gemma4 on Xeon

合并时间：2026-08-17 10:52

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/22498

执行摘要

- 一句话：为 Xeon CPU 补齐 Gemma4 全链路推理支持
- 推荐动作：值得精读。重点看三个设计决策：一是 KV-shared 层如何绕过 is_cross_attn 的语义错配、在 stage 1 内完成因果掩码并处理空行清零；二是跨平台内核 API 变更时如何同步 x86 / aarch64 双实现并显式拒绝不支持路径；三是 review 驱动的 " 剪枝 "——及时移除 TP3/6 这种过度工程化的 workaround，保持 CPU 配置逻辑集中在 update_config.py 一处。

功能与动机

PR body 明确说明目标："This PR aims to add support for Gemma4 on Xeon"，基本功能部分移植了 GPT-OSS 的 PR #16775（sliding window attention 支持与 MoE 接口更新），并补充 MoE Gelu 激活、TP 支持以及多项 CPU 性能优化（weight_packed_linear 应用于 lm_head、flash_attn_varlen_func、多维 RoPE 内核）。

实现拆解

实现按以下 5 个步骤拆解：

1. 模型与权重加载层：python/sglang/srt/models/gemma4_causal.py 中 Gemma4Attention 改为对 sliding_attention 层读取独立的 swa_num_attention_heads / swa_num_key_value_heads 配置，因为 full-attention 层按 num_attention_heads 做的 padding 对滑动层不适用；同时提取 load_tied_lm_head() 供 pp_filter_load_weight 与多模态模型共用。python/sglang/srt/models/gemma4_mm.py 新增 lm_head_is_tied 判定：CPU + AMX 场景下 packed head 权重无法与 embedding 表 alias，必须物化 ParallelLMHead 并在 load_weights 时把 embed_tokens.weight 复制进 lm_head。
2. sgl-kernel CPU 内核层：python/sglang/kernels/aot/csrc/cpu/rope.cpp 新增 apply_multidimensional_rope_cpu（按 ndim=2 分块、每块独立应用 rotary，含 AVX512 半宽向量路径并融合 q/k 处理），并把 multimodal_rotary_embedding_cpu 从返回 tuple 改为原地写入；moe.cpp / gemm.h / moe_fp8.cpp 为 fused_experts_cpu 增加 activation 参数支持 gelu_and_mul，aarch64/moe.cpp 同步新签名但仅允许 silu；extend.cpp 增加 kv_from_cache 分支，为 KV-shared 层在 stage 1 内完成因果掩码。
3. 注意力后端子层：python/sglang/srt/layers/attention/intel_amx_backend.py 按 (num_heads, v_head_dim) 键缓存 decode 所需的 attn_logits buffer，避免 Gemma4 混

合 head_dim 的层在每次 decode 时重新分配；forward_extend 返回改为 o.view(-1, ...) 并支持 k=v=None 的 KV-shared 层。python/sglang/srt/model_executor/cpu_graph_runner.py 把 multimodal_rotary_embedding_cpu 与 apply_multidimensional_rope_cpu 注册为 none-return fake op，供 torch.compile 使用。

4. 配置层：python/sglang/srt/configs/update_config.py 新增 adjust_swa_num_heads_if_necessary 统一 sliding-window 层的 head padding 逻辑；修复 attr_value is None 时取模崩溃；多模态配置遍历增加 None 保护。
5. 测试配套：test/registered/cpu/test_rope.py 新增 test_apply_multidimensional_rope（覆盖 head_dim=160 的标量尾部边界、bf16/fp16 与 sincos dtype 组合）；test/registered/cpu/test_moe.py 为 bf16/fp8/mxfp4 参数化 gelu 激活并新增 test_unsupported_activation_is_rejected；test/registered/cpu/test_extend.py 新增 test_extend_attention_kv_from_cache 验证缓存读取模式的掩码正确性。

关键文件：

- python/sglang/srt/models/gemma4_causal.py（模块 模型层；类别 source；类型 data-contract；符号 load_tied_lm_head, pp_filter_load_weight, Gemma4Attention）：Gemma4 文本模型核心改动：sliding_attention 层改用独立的 swa head 配置，提取 load_tied_lm_head 供 PP 与多模态路径复用。
- python/sglang/kernels/aot/csrc/cpu/extend.cpp（模块 内核；类别 source；类型 core-logic；符号 extend_attention_cpu, extend_attention_kernel_impl）：新增 kv_from_cache 分支修复 KV-shared 层的因果掩码正确性问题：此前借道 is_cross_attn 路径导致层会关注到自己的未来 token。
- python/sglang/srt/layers/attention/intel_amx_backend.py（模块 注意力；类别 source；类型 core-logic；符号 _get_attn_logits_buffer, forward_decode, forward_extend）：decode 路径按层形状缓存 attn_logits buffer，修复 Gemma4 混合 head_dim 层每次 decode 重复分配零初始化 buffer 的性能问题。
- python/sglang/srt/configs/update_config.py（模块 配置层；类别 source；类型 core-logic；符号 adjust_swa_num_heads_if_necessary, adjust_config_with_unaligned_cpu_tp）：提取 adjust_swa_num_heads_if_necessary 统一 sliding-window 层 head padding，并修复 attr_value is None 时的取模崩溃。
- python/sglang/srt/models/gemma4_mm.py（模块 模型层；类别 source；类型 data-contract；符号 Gemma4MultimodalEmbedder, load_weights, lm_head_is_tied）：多模态入口在 CPU AMX 下关闭 lm_head 与 embedding 的模块别名，改由 load_weights 复制权重，是 CPU 支持的关键数据契约变更。
- python/sglang/kernels/aot/csrc/cpu/rope.cpp（模块 内核；类别 source；类型 core-logic；符号 apply_multidimensional_rope_cpu, multimodal_rotary_embedding_cpu）：新增 apply_multidimensional_rope_cpu 内核，支撑 Gemma4 视觉塔的二维 RoPE，并优化 AVX512 路径。
- test/registered/cpu/test_extend.py（模块 测试；类别 test；类型 test-coverage；符号 test_extend_attention_kv_from_cache）：新增 test_extend_attention_kv_from_cache 覆盖缓存读取模式的因果掩码，是本次正确性修复的回归保障。

- test/registered/cpu/test_rope.py（模块 测试；类别 test；类型 test-coverage；符号 test_apply_multidimensional_rope, _apply_multidimensional_rope_ref）：新增 test_apply_multidimensional_rope 覆盖新内核的向量 / 标量尾部边界，含 head_dim=160 的尾块场景。
- test/registered/cpu/test_moe.py（模块 测试；类别 test；类型 test-coverage；符号 test_unsupported_activation_is_rejected, test_bf16_moe, test_fp8_moe, test_mxfp4_moe）：为 bf16/fp8/mxfp4 MoE 参数化 gelu 激活并新增不支持激活的拒绝测试，覆盖新 activation 参数。

关键符号：load_tied_lm_head, pp_filter_load_weight, adjust_swa_num_heads_if_necessary, _get_attn_logits_buffer, apply_multidimensional_rope_cpu, extend_attention_cpu, lm_head_is_tied

关键源码片段

python/sglang/srt/models/gemma4_causal.py

Gemma4 文本模型核心改动：sliding_attention 层改用独立的 swa head 配置，提取 load_tied_lm_head 供 PP 与多模态路径复用。

```
# 从 pp_filter_load_weight 中提取的共享 tied lm_head 加载器：
# 当运行时无法通过模块别名完成 embedding 与 lm_head 的权重绑定
#  （PP 下 embed 在首 rank、lm_head 在末 rank；CPU AMX 下 lm_head
# 使用 packed weight，无法与 embedding 表 alias）时，把 checkpoint
# 里的 embed_tokens.weight 显式加载进 lm_head。
def load_tied_lm_head(
    loaded_weight, *, params_dict, loaded_params, head_param_name="lm_head.weight"
):
    head_param = params_dict.get(head_param_name)
    if head_param is None:
        # 本 rank 不持有 lm_head（如非末 PP rank），直接跳过
        return
    wl = getattr(head_param, "weight_loader", default_weight_loader)
    wl(head_param, loaded_weight)
    loaded_params.add(head_param_name)


# Gemma4Attention 的 head 数解析：sliding_attention 层与 full attention
# 层携带不同的 head 数配置（swa_num_attention_heads 等），TP 切分必须
# 按各自配置分别校验，否则滑动层会因 head 数不可分而越界。
layer_type = config.layer_types[layer_id]
if layer_type == "sliding_attention":
    self.total_num_heads = getattr(
        config, "swa_num_attention_heads", config.num_attention_heads
    )
    self.total_num_kv_heads = getattr(
        config, "swa_num_key_value_heads", config.num_key_value_heads
    )
else:
```

```
        self.total_num_heads = config.num_attention_heads
        self.total_num_kv_heads = config.num_key_value_heads
    assert self.total_num_heads % tp_size == 0
    self.num_heads = self.total_num_heads // tp_size
```

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

decode 路径按层形状缓存 attn_logits buffer，修复 Gemma4 混合 head_dim 层每次 decode 重复分配零初始化 buffer 的性能问题。

```
    def forward_decode(self, q, k, v, layer, forward_batch, save_kv_cache=True, sinks=None):
        if self.draft_decode_metadata is not None:
            req_to_token, seq_lens, req_pool_indices = self.draft_decode_metadata
        else:
            req_to_token = self.req_to_token_pool.req_to_token
            req_pool_indices = forward_batch.req_pool_indices
            seq_lens = forward_batch.seq_lens

        q = q.reshape(-1, layer.tp_q_head_num * layer.qk_head_dim)
        seq_lens = forward_batch.seq_lens
        if seq_lens.dtype != torch.int64:
            seq_lens = seq_lens.to(torch.int64)

        # Gemma 4 的 sliding attention 层与 full attention 层使用不同的
        # head 数与 head_dim，模型级 metadata buffer 只对其中一种形状匹配；
        # 另一种形状若直接复用会越界，这里按层形状取专用 buffer
        if layer.v_head_dim == self.v_head_dim and layer.tp_q_head_num == self.num_head:
            attn_logits, _ = self.forward_metadata
        else:
            attn_logits = self._get_attn_logits_buffer(
                seq_lens.shape[0], layer.tp_q_head_num, layer.v_head_dim
            )
        # ... 调用 decode_attention_fwd ...
        return o.view(-1, layer.tp_q_head_num * layer.v_head_dim)


    def _get_attn_logits_buffer(self, num_seqs, num_heads, v_head_dim):
        # 以 (num_heads, v_head_dim) 为键缓存；decode_attention_cpu 会写入它
        # 随后读取的每个元素，所以 buffer 无需初始化、可跨步复用，只增不减
        key = (num_heads, v_head_dim)
        buffer = self._attn_logits_buffers.get(key)
        if buffer is None or buffer.shape[0] < num_seqs:
            buffer = torch.empty(
                (num_seqs, num_heads, self.num_kv_splits, v_head_dim + 1),
                dtype=torch.float32,
                device=self.device,
            )
            self._attn_logits_buffers[key] = buffer
        return buffer[:num_seqs]
```

python/sglang/srt/configs/update_config.py

提取 adjust_swa_num_heads_if_necessary 统一 sliding-window 层 head padding，并修复 attr_value is None 时的取模崩溃。

```
def adjust_swa_num_heads_if_necessary(model_config, tp_size, weight_block_size):
    # Sliding-window 层携带独立的 head 数配置，full-attention 层按
    # num_attention_heads 做的 padding 对它们不适用，必须按 swa_head_dim
    # 单独 padding，否则 TP 切分时滑动层会因 head 数不可分而越界
    from sglang.srt.layers.vocab_parallel_embedding import pad_vocab_size

    text_config = model_config.hf_text_config
    if not hasattr(text_config, "swa_num_key_value_heads"):
        return

    swa_num_key_value_heads = text_config.swa_num_key_value_heads
    swa_num_attention_heads = getattr(
        text_config, "swa_num_attention_heads", model_config.num_attention_heads
    )
    # ModelConfig 总是物化 swa_head_dim，默认与 head_dim 一致
    swa_pad_size = get_num_heads_padding_size(
        tp_size, weight_block_size, text_config.swa_head_dim
    )
    padded_num_key_value_heads = pad_vocab_size(swa_num_key_value_heads, swa_pad_size)
    padded_num_attention_heads = padded_num_key_value_heads * (
        swa_num_attention_heads // swa_num_key_value_heads
    )

    update_config(text_config, "swa_num_key_value_heads", padded_num_key_value_heads)
    update_config(text_config, "swa_num_attention_heads", padded_num_attention_heads)
```

评论区精华

Review 中最有价值的交锋集中在三处：一是 maintainer mingfeima 认为 TP3/6 的 head padding workaround 太特殊（"gemma4 is a series of small LLMs"），要求全部移除并留作未来 TODO，htzo 随后提交 "gemma4: Remove the TP 3 and 6 workarounds"；二是 SWA head padding 逻辑最初分散在三个 config 对象循环中且只写 hf_text_config，被指出 " 不是很符合逻辑 " 后重构为独立的 adjust_swa_num_heads_if_necessary()；三是 KV-shared 层原本借道 is_cross_attn 的路径被 htzo 在提交说明中揭露会 "attend to their own future"，最终通过 kv_from_cache 参数在 stage 1 内完成因果掩码。另有对 decode.cpp 中 i % num_heads 整数除法性能的提醒，以及 log_debug_on_rank0 是否应作为独立 PR 的讨论（最终保留但未拆出）。

- KV-shared 层因果掩码正确性 (correctness): extend_attention_cpu 新增 kv_from_cache 分支，在 stage 1 内完成因果掩码；intel_amx_backend 按 mingfeima 建议提取 is_extend_cache_read_only 命名该条件并补充文档。
- TP3/6 workaround 是否保留 (design): htzo 提交 "gemma4: Remove the TP 3 and 6 workarounds" 删除相关代码，并顺带移除了 default_weight_loader 中仅为 TP3/6 服务的

partial copy 逻辑。

- SWA head padding 逻辑抽象 (design): htzo 提取 adjust_swa_num_heads_if_necessary()，与 adjust_tp_num_heads_if_necessary() 并列，统一通过 update_config 更新 swa 配置。
- fused_experts_cpu 新增 activation 参数破坏 aarch64 ABI (correctness): htzo 提交 "sgl-kernel: Add activation arg to aarch64 fused_experts_cpu"，aarch64 仅支持 silu，其他激活显式报错。
- decode_accumulate_kv_splits 中取模性能 (performance): 该评论指向 sgl-kernel 独立仓库代码，本 PR 提交列表中未见对应修复，可能遗留待后续处理。
- log_debug_on_rank0 的归属 (design): 函数保留在 update_config.py 中使用，未拆分为独立 PR，但该工具函数本身实现简单、侵入面小。

风险与影响

- 风险：主要风险集中在四方面：
  1. extend.cpp 内核通用路径变更：kv_from_cache 是 CPU extend attention 的通用改动，虽只在新分支生效，但 kv_end 计算与空行清零逻辑位于共享循环内，非 Gemma4 模型的回归风险需依靠 CI 覆盖；
  2. MoE 内核 API 签名变更：fused_experts_cpu 增加 activation 参数贯穿 x86 与 aarch64 两条实现，aarch64 目前仅支持 silu，其他激活会 TORCH_CHECK 拒绝，未来若在 ARM 上跑 Gemma4 会直接失败；
  3. attn_logits buffer 复用的隐含契约：_get_attn_logits_buffer 依赖 decode_attention_cpu " 写入每个随后读取的元素 " 这一隐含约定，一旦内核改为读取未初始化元素将产生随机错误；
  4. CPU 下 tie_word_embeddings 语义变化：AMX 场景下 lm_head 不再 alias embed_tokens，get_embed_and_head() 返回两个独立 tensor 的语义依赖 load_weights 的复制路径，RL 权重同步等调用方需留意。 - 影响：对用户而言，Intel Xeon（AMX）用户首次可以在 CPU 上完整运行 Gemma4 文本与多模态模型，且 MoE 支持 bf16/fp8/mxfp4 多种量化配合 Gelu 激活；对系统而言，新增 apply_multidimensional_rope_cpu 与 extend_attention_cpu 的 kv_from_cache 契约，CPU AOT 内核需要同步编译注册；对团队而言，CPU 后端维护者需要同时维护 full-attention 与 SWA 两套 head 配置（num_attention_heads vs swa_num_attention_heads），并保持 x86 / aarch64 内核签名一致。 - 风险标记：核心内核路径变更，跨平台 ABI 同步风险，注意力掩码正确性，隐含 buffer 复用契约

关联脉络

- PR #16775 GPT-OSS（移植来源）：PR body 明确说明本 PR 移植了该 PR 的 sliding window attention 支持（intel_amx_attn、torch_native 后端）与 MoE 相关接口更新，是本次功能的基础。
- PR #25220 （被指破坏 Xeon CI）：htzo 在 issue 评论中指出 Xeon-gnr 的 CI 因 #25220 失败，本 PR 的 CI 绿灯依赖该问题修复后重跑。