PR #30716 完整报告

sgl-project/sglang

[Diffusion] Revert CPU AMX optimizations

合并时间：2026-07-10 09:09

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/30716

执行摘要

- 一句话：回滚 CPU AMX 优化，修复 FSDP 推理崩溃
- 推荐动作：值得精读以理解 FSDP 加载器中 process_weights_after_loading 和 device_loading_context 的正确使用模式。设计决策关注：量化钩子必须幂等；设备上下文管理在 FSDP 下需谨慎。后续开发者应以此为鉴，测试需覆盖多 GPU + 量化场景。

功能与动机

PR body 指出：#28527 加入的第二次无条件 process_weights_after_loading() 对于量化权重不是幂等的，并且 device_loading_context 在 FSDP 推理中尝试将 CUDA 支持的参数恢复为 CPU 存储，导致存储不匹配。CI 任务 multimodal-gen-test-2-gpu 因 device_loading_context 中的 CPU/CUDA 存储不匹配而失败。因此需要回滚以稳定 CI。

实现拆解

1. 移除 AMX 注意力后端：删除 python/sglang/multimodal_gen/runtime/layers/attention/backends/amx_attn.py 整个文件，该文件包含 AMXAttentionBackend 类和新 AMXATTNImpl 实现，两者均被移除。
2. 清理线性层 AMX 集成：在 python/sglang/multimodal_gen/runtime/layers/linear.py 的 UnquantizedLinearMethod 中，删除 process_weights_after_loading 方法中对 _amx_process_weight_after_loading 的调用，并移除 apply 方法中使用 weight_packed_linear 的 AMX 分支，回归到标准的 F.linear。
3. 回退 CPU 平台注意力后端选择：在 python/sglang/multimodal_gen/runtime/platforms/cpu.py 的 get_attn_backend_cls_str 中，移除对 AMX_ATTN 后端的支持，只返回 Torch SDPA 后端；同时删除了导入 cpu_has_amx_support 和相关检测模块。
4. 修复 FSDP 加载器中的重复后处理：在 fsdp_load.py 和 text_encoder_loader.py 中，移除由源 PR 添加的第二次 process_weights_after_loading 调用循环，该循环使用了 device_loading_context，导致 FSDP 推理中的设备上下文不匹配。现在权重后处理仅进行一轮。
5. 内核与测试配套调整：在 sgl-kernel/csrc/cpu/flash_attn.cpp 中，将 sm_scale 从可选参数改回必选（还原为固定默认值 1/sqrt(head_size)）；在 test/registered/cpu/test_flash_attn.py 中移除 AMX 特定测试用例；其他文件如 vision.py、interface.py、wanvae.py、vae_loader.py 等也移除了对 AMX 相关的引用或配置。

关键文件：

- python/sglang/multimodal_gen/runtime/layers/attention/backends/amx_attn.py（模块 注意力后端；类别 source；类型 deletion；符号 AMXAttentionBackend, get_supported_head_sizes, get_enum, get_impl_cls）：被完全删除的 AMX 注意力后端文件，是整个回滚的核心。
- python/sglang/multimodal_gen/runtime/layers/linear.py（模块 线性层；类别 source；类型 core-logic；符号 process_weights_after_loading, apply）：移除了线性层中 AMX 权重后处理和 AMX 感知的 apply 方法，回归标准计算。
- python/sglang/multimodal_gen/runtime/platforms/cpu.py（模块 平台路由；类别 source；类型 dependency-wiring；符号 get_attn_backend_cls_str）：CPU 平台注意力后端选择逻辑移除 AMX 选项，始终使用 Torch SDPA。
- python/sglang/multimodal_gen/runtime/loader/fsdp_load.py（模块 FSDP 加载；类别 source；类型 dependency-wiring；符号 maybe_load_fsdp_model, device_loading_context）：移除了导致 FSDP 推理失败的二次 process_weights_after_loading 循环和 device_loading_context。
- python/sglang/multimodal_gen/runtime/loader/component_loaders/text_encoder_loader.py（模块 文本编码加载；类别 source；类型 core-logic；符号 load_model）：同步移除另一处 device_loading_context 包裹的二次 process_weights_after_loading。
- sgl-kernel/csrc/cpu/flash_attn.cpp（模块 CPU 内核；类别 source；类型 core-logic；符号 flash_attn_varlen_func）：恢复 sm_scale 参数为必选，与删除 AMX 后端适配。
- test/registered/cpu/test_flash_attn.py（模块 CPU 测试；类别 test；类型 test-coverage）：移除 AMX 相关的测试用例，与源码删除保持一致。

关键符号：AMXAttentionBackend, AMXATTNImpl, UnquantizedLinearMethod.process_weights_after_loading, UnquantizedLinearMethod.apply, CpuPlatform.get_attn_backend_cls_str, flash_attn_varlen_func, maybe_load_fsdp_model, device_loading_context

关键源码片段

python/sglang/multimodal_gen/runtime/layers/attention/backends/amx_attn.py

被完全删除的 AMX 注意力后端文件，是整个回滚的核心。

```
# 文件 amx_attn.py（已删除）
# 该文件实现了 CPU AMX 注意力后端，基于 flash_attn_varlen_func
import torch
from sglang.multimodal_gen.runtime.layers.attention.backends.attention_backend import (
    AttentionBackend, AttentionImpl, AttentionMetadata,
)
from sglang.multimodal_gen.runtime.platforms import AttentionBackendEnum

logger = init_logger(__name__)
flash_attn_varlen_func = torch.ops.sgl_kernel.flash_attn_varlen_func
```

```
class AMXAttentionBackend(AttentionBackend):
    accept_output_buffer: bool = True

    @staticmethod
    def get_supported_head_sizes() -> list[int]:
        return [32, 64, 96, 128, 160, 192, 224, 256]

    @staticmethod
    def get_enum() -> AttentionBackendEnum:
        return AttentionBackendEnum.AMX_ATTN

    @staticmethod
    def get_impl_cls() -> type["AMXATTNImpl"]:
        return AMXATTNImpl

class AMXATTNImpl(AttentionImpl):
    def __init__(self, num_heads, head_size, causal, softmax_scale, num_kv_heads=None, prefix=
    "", **extra_impl_args):
        self.causal = causal
        self.softmax_scale = softmax_scale

    def forward(self, query, key, value, attn_metadata):
        # 仅支持 batch=1 场景，直接调用 flash_attn_varlen_func
        max_seqlen_q = query.shape[1]
        max_seqlen_k = key.shape[1]
        return flash_attn_varlen_func(
            query[0], key[0], value[0],
            torch.tensor([0, max_seqlen_q]).to(torch.int),
            torch.tensor([0, max_seqlen_k]).to(torch.int),
            max_seqlen_q, max_seqlen_k,
            self.causal, self.softmax_scale,
        ).unsqueeze(0)
```

python/sglang/multimodal_gen/runtime/layers/linear.py

移除了线性层中 AMX 权重后处理和 AMX 感知的 apply 方法，回归标准计算。

```
# 文件 linear.py（变更后关键部分）
class UnquantizedLinearMethod(LinearMethodBase):
    def create_weights(self, layer, input_size_per_partition, output_partition_sizes,
                       input_size, output_size, params_dtype, **extra_weight_attrs):
        weight = Parameter(
            torch.empty(sum(output_partition_sizes), input_size_per_partition, dtype=params_dtype)
            ,
            requires_grad=False,
        )
        set_weight_attrs(weight, {"input_dim": 1, "output_dim": 0})
        layer.register_parameter("weight", weight)
        set_weight_attrs(weight, extra_weight_attrs)

    # process_weights_after_loading 已被完全删除，不再调用 _amx_process_weight_after_loading
```

```
    # 之前存在的 AMX 重打包逻辑已经移除

    def apply(self, layer, x, bias=None):
        # 移除了 use_intel_amx_backend 分支，直接使用标准 F.linear
        output = (
            F.linear(x, layer.weight, bias)
            if IS_AMP_SUPPORTED or bias is None
            else F.linear(x, layer.weight, bias.to(x.dtype))
        )
        return output
```

python/sglang/multimodal_gen/runtime/platforms/cpu.py

CPU 平台注意力后端选择逻辑移除 AMX 选项，始终使用 Torch SDPA。

```
# 文件 cpu.py（变更后关键部分）
    @classmethod
    def get_attn_backend_cls_str(cls, selected_backend, head_size, dtype):
        # 移除了对 AMX_ATTN 的处理，只接受 TORCH_SDPA 或 None
        if selected_backend not in (None, AttentionBackendEnum.TORCH_SDPA):
            logger.warning(
                "%s is not supported on CPU; falling back to Torch SDPA.",
                selected_backend,
            )
        logger.info("Using Torch SDPA backend for CPU.")
        return (
            "sglang.multimodal_gen.runtime.layers.attention.backends.sdpa.SDPABackend"
        )
```

评论区精华

本 PR 无实质性 review 讨论。作者 mickqian 在评论中要求合并前确保 multimodal-gen 测试通过（@mingfeima, *please make sure all nvidia multimodal-gen tests pass before merging*），BBuf 直接批准合并。另有两个 bot 自动评论提示配额限制。

- 确保 CI 测试通过 (testing): BBuf 在 CI 通过后批准合并。

风险与影响

- 风险：回滚后 CPU 扩散推理将失去 AMX 加速，性能回退到 Torch SDPA 水平。但原 AMX 实现本身存在功能缺陷，回滚消除了这些风险。需确认无遗漏对已删除符号（如 AMXAttentionBackend、_amx_process_weight_after_loading）的引用，否则可能引发 ImportError。当前变更加盖了所有导入点和调用点，风险较低。
- 影响：对用户：CPU 上运行扩散模型的用户将不再享受 AMX 加速，推理速度可能下降。对系统：FSDP 分布式推理恢复到稳定状态，量化权重加载正确。对团队：短期内需重新设计 AMX 集成方案，确保与 FSDP 和量化兼容。影响范围限定于 diffusion 子系统的 CPU 后端，不涉及 GPU 路径。
- 风险标记：FSDP 兼容性修复，量化权重幂等性，多 GPU 回归防护

关联脉络

- PR #28527 [Diffusion][CPU] Adding AMX optimizations for CPU platform: 被本回滚 PR 的源 PR，引入 FSDP 推理破坏性变更。