PR #52616 完整报告

vllm-project/vllm

[CPU] Add AMX-only high-performance MLA backend for DeepSeek V2/V3/R1

合并时间：2026-08-19 10:28

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/52616

执行摘要

- 一句话：新增 CPU AMX 高性能 MLA 后端并修复共享 MLA 两个 bug
- 推荐动作：值得精读。建议重点看三点：(1) AMXMLAMetadataBuilder.build 把每 step 不变的张量计算提升到 builder 一次完成，避免逐层重复，是 CPU 后端性能设计的典型套路；(2) mla_attention.py 中 is_amx_bmm_enabled 分支与 process_weights_after_loading 委托 impl 的模式，可平滑扩展到其他自研 bmm 后端；(3) commit 历史中关于 fused_qkv_rope 被丢弃的排查过程，展示了 torch.compile 与自定义 op 融合的兼容性陷阱，对后续 CPU 算子设计有直接借鉴价值。

功能与动机

PR body 明确说明动机：已有参考实现 CPUMLABackend（#49453）的目标是任意 CPU/dtype、block_size=16 的"functional/CI reference, not performance"，而本 PR 为 CPU 提供真正的高性能 MLA 路径，要求 AMX（bf16、block_size 为 32 的倍数），并由平台层在主机支持时自动优选、否则回退。PR 顺带修复了两个在接入高性能路径过程中暴露的共享 bug：一是 CPU 通用 Linear 权重打包路径在 MLAAttention.process_weights_after_loading 读取 raw weight 推导 W_UK/W_UV 之前就释放了 kv_b_proj 的原始权重；二是 get_mla_prefill_backend() 在 device_capability 为 None 时无条件解析 CUDA-only 的 FlashAttnPrefillBackend，在 CPU 上构造即断言。issue 评论中 maobaolong 也确认其 PR #49453 无法通过 vllm + dsv2 的 e2e 测试，本 PR 修复了该问题并在其本地 macOS/Ubuntu 环境验证通过。

实现拆解

1. 算子与构建接入

- vendored SGLang 的 csrc/cpu/sgl-kernels/（decode.cpp 1821 行、extend.cpp 568 行、bmm.cpp 208 行、mla_cache.cpp 111 行、flash_attn.h 250 行及 qkv_proj 相关），在 cmake/cpu_extension.cmake 中用 __AVX512BF16__&& __AVX512F__&& __AVX512VNNI__ 门控，仅 AMX 目标编译。
- csrc/cpu/torch_bindings.cpp 注册 decode_attention_cpu/extend_attention_cpu/bmm_cpu/concat_and_cache_mla_cpu 四个算子，vllm/_custom_ops.py 新增 cpu_mla_decode、cpu_mla_extend、bmm_cpu、amx_mla_concat_and_cache 四个 Python wrapper 供后端调用。

2. AMX MLA 后端实现

- 新增 vllm/v1/attention/backends/mla/amx_mla.py：AMXMLABackend 声明 bf16、head_size=576（kv_lora_rank 512 + qk_rope_head_dim 64）、block_size 为 32 的倍数；AMXMLAMetadataBuilder.build 每 step 一次完成 block-table 展开、seq_lens 的 int64 转换、req_pool_indices、num_kv_splits 及 extend 段信息推导；AMXMLAImpl.process_weights_after_loading 从 kv_b_proj 推导并用 convert_weight_packed VNNI 打包 W_UK/W_UV；do_kv_cache_update 改用 amx_mla_concat_and_cache；forward_mqa 只调用 cpu_mla_decode；forward_mha 完全覆盖，直接用 extend_attention_cpu 在 latent-MQA 空间一次因果 pass 完成 fresh prefill 与 cached-prefix continuation。

3. 共享 MLA/Linear 代码修复与 AMX bmm 复用

- vllm/model_executor/layers/linear.py 增加 _cpu_keep_raw_weight 标记，阻止 CPU 打包路径提前释放 kv_b_proj 的 raw weight。
- mla_attention.py 增加 is_amx_bmm_enabled 分支：decode 的 Q 吸收 / 解吸复用 AMX 打包权重（bmm_cpu），并委托 self.impl.process_weights_after_loading；AMX 启用后释放 kv_b_proj 原始权重。

4. 平台选择与 prefill 后端注册

- vllm/platforms/cpu.py 根据 torch.cpu._is_amx_tile_supported() 自动选择 AMX_MLA，并豁免 AMX 路径的 block_size/chunked-prefill 强制配置。
- 新增 prefill/cpu_native.py 的 CPUNativeMLAPrefillBackend 占位类，prefill/selector.py 修正 device_capability is None 时的分支，prefill/registry.py 与 v1/attention/backends/registry.py 注册 CPU_NATIVE 与 AMX_MLA。

5. 测试与 CI 配套

- 新增 tests/kernels/attention/test_amx_mla.py（kernel 级 bmm/decode/extend/cache round-trip 对照纯 PyTorch 参考，后端级 forward_mqa/forward_mha 对照），门控 torch.cpu._is_amx_tile_supported()，并接入 .buildkite/hardware_tests/cpu.yaml。
- 提交演进中包含 4 次 metadata hoist 优化、fused_qkv_rope 路径因 torch.compile 崩溃被丢弃、decode 复用 AMX bmm、CI 回归修复等关键决策。

关键文件：

- vllm/v1/attention/backends/mla/amx_mla.py（模块 MLA 后端；类别 source；类型 core-logic；符号 _compute_num_kv_splits, _expand_block_table, AMXMLABackend, AMXMLAMetadataBuilder）：新增 AMX_MLA 后端核心文件，包含 AMXMLABackend/AMXMLAImpl/AMXMLAMetadataBuilder，是整个 PR 的入口与主逻辑。
- vllm/model_executor/layers/attention/mla_attention.py（模块 注意力层；类别 source；类型 data-contract）：共享 MLA 核心逻辑被修改，新增 is_amx_bmm_enabled 分支与 impl 权重后处理委托，影响所有 MLA 后端。
- csrc/cpu/sgl-kernels/decode.cpp（模块 CPU 内核；类别 source；类型 core-logic）：vendored SGLang AMX decode kernel，1821 行，包含 VNNI 打包与 bf16 矩阵乘主循

环，是 decode 性能核心。

- csrc/cpu/sgl-kernels/extend.cpp（模块 CPU 内核；类别 source；类型 core-logic）：vendored SGLang AMX extend kernel，分两阶段处理 prefix 与 new tokens 的因果注意力，是 prefill 性能核心。
- vllm/_custom_ops.py（模块 自定义算子；类别 source；类型 core-logic；符号 cpu_mla_decode, cpu_mla_extend, bmm_cpu, amx_mla_concat_and_cache）：新增 cpu_mla_decode、cpu_mla_extend、bmm_cpu、amx_mla_concat_and_cache 四个 Python wrapper，是 Python 层到 C++ op 的桥接。
- csrc/cpu/sgl-kernels/mla_cache.cpp（模块 KV 缓存写入；类别 source；类型 core-logic）：实现 CPU 版 concat_and_cache_mla，将 kv_c_normed 与 k_pe 写入同一 576 宽 latent cache 行，补齐 GPU-only 算子在 CPU 的缺口。
- vllm/platforms/cpu.py（模块 平台选择；类别 source；类型 core-logic）：平台层决定是否自动选择 AMX_MLA，是后端开关与回退策略的关键入口。
- vllm/model_executor/layers/linear.py（模块 权重打包；类别 source；类型 data-contract）：新增 _cpu_keep_raw_weight 标记修复 kv_b_proj 原始权重被提前释放的共享 bug，是可复用的防御性改动。
- vllm/v1/attention/backends/mla/prefill/cpu_native.py（模块 prefill 注册；类别 source；类型 dependency-wiring；符号 CPUNativeMLAPrefillBackend, get_name, run_prefill_new_tokens, run_prefill_context_chunk）：新增 inert CPU prefill backend 占位类，满足 MLA 抽象的结构要求，修复 CPU 上 prefill backend 解析断言。
- vllm/v1/attention/backends/mla/prefill/selector.py（模块 prefill 选择；类别 source；类型 core-logic）：修正 device_capability 为 None 时的 CPU 分支判定位置，避免 mock 干扰导致 GPU 测试误入 CPU 分支。
- tests/kernels/attention/test_amx_mla.py（模块 测试覆盖；类别 test；类型 test-coverage；符号 _flatten_cache, _random_paged_cache, _ref_latent_attn, test_bmm_cpu_matches_torch_bmm）：新增 471 行 AMX MLA 正确性测试，覆盖 bmm/decode/extend/cache 与后端级 forward 对照，是质量保障核心。
- csrc/cpu/torch_bindings.cpp（模块 算子绑定；类别 source；类型 core-logic）：注册四个 CPU MLA 相关 C++ op，是 Python wrapper 与内核实现之间的绑定层。
- cmake/cpu_extension.cmake（模块 构建配置；类别 other；类型 configuration）：构建门控决定 vendored kernels 是否编译，是 AMX-only 特性的基础设施开关。

关键符号：AMXMLABackend.get_supported_head_sizes, AMXMLABackend.supports_block_size, AMXMLABackend.get_name, AMXMLAMetadataBuilder.build, AMXMLAImpl.process_weights_after_loading, AMXMLAImpl.do_kv_cache_update, AMXMLAImpl.forward_mqa, AMXMLAImpl.forward_mha, _compute_num_kv_splits, _expand_block_table, CPUNativeMLAPrefillBackend.get_name, CPUNativeMLAPrefillBackend.run_prefill_new_tokens, CPUNativeMLAPrefillBackend.run_prefill_context_chunk, MLAAttention.process_weights_after_loading, MLAAttention.forward_impl, concat_and_cache_mla_cpu, bmm_cpu

关键源码片段

vllm/v1/attention/backends/mla/amx_mla.py

新增 AMX_MLA 后端核心文件，包含 AMXMLABackend/AMXMLAImpl/AMXMLAMetadataBuilder，是整个 PR 的入口与主逻辑。

```
class AMXMLAMetadataBuilder(MLACommonMetadataBuilder[MLACommonMetadata]):
    def build(self, common_prefix_len, common_attn_metadata, fast_build: bool = False):
        # 把每 step 不变的计算收敛到这里做一次，避免 decode/prefill 在每个
        # layer 的 forward_mqa/forward_mha 里重复做同样的张量算术。
        attn_metadata = super().build(
            common_prefix_len, common_attn_metadata, fast_build
        )
        block_size = self.kv_cache_spec.block_size

        # decode 侧：vLLM 的 block-table 分页展开成 kernel 需要的扁平物理行索引，
        # 同时补上 seq_lens 的 int64 版本与 request 池索引。
        if attn_metadata.decode is not None:
            decode = attn_metadata.decode
            decode.req_to_token = _expand_block_table( # type: ignore[attr-defined]
                decode.block_table, block_size
            )
            decode.seq_lens_i64 = decode.seq_lens.to(torch.int64) # type: ignore[attr-defined]
            decode.req_pool_indices = torch.arange( # type: ignore[attr-defined]
                decode.block_table.size(0),
                dtype=torch.int64,
                device=decode.block_table.device,
            )
            # 用 CPU 线程数代替 GPU SM 数估算 KV split 数，规则对齐 TritonMLAImpl。
            decode.num_kv_splits = _compute_num_kv_splits( # type: ignore[attr-defined]
                attn_metadata.max_seq_len, current_platform.num_compute_units()
            )

        # prefill 侧：extend kernel 需要每个 request 的总上下文长度（prefix + new tokens），
        # 以及扩展段的起始位置 / 长度；这些都能从 query_start_loc 一次推导并缓存。
        if attn_metadata.prefill is not None:
            prefill = attn_metadata.prefill
            num_decodes = attn_metadata.num_decodes
            num_prefills = attn_metadata.num_prefills
            prefill.cpu_seq_lens = common_attn_metadata.seq_lens[ # type: ignore[attr-defined]
                num_decodes : num_decodes + num_prefills
            ].to(torch.int64)
            prefill.req_to_token = _expand_block_table( # type: ignore[attr-defined]
                prefill.block_table, block_size
            ).to(torch.int64)
            prefill.req_pool_indices = torch.arange( # type: ignore[attr-defined]
                prefill.block_table.size(0),
                dtype=torch.int64,
                device=prefill.block_table.device,
```

```
            )
            query_start_loc_i64 = prefill.query_start_loc.to(torch.int64)
            extend_seq_lens = query_start_loc_i64[1:] - query_start_loc_i64[:-1]
            prefill.extend_seq_lens = extend_seq_lens # type: ignore[attr-defined]
            prefill.extend_start_loc = query_start_loc_i64[:-1] # type: ignore[attr-defined]
            prefill.max_len_extend = int(extend_seq_lens.max().item()) # type: ignore[attr-defined]
        return attn_metadata
```

vllm/model_executor/layers/attention/mla_attention.py

共享 MLA 核心逻辑被修改，新增 is_amx_bmm_enabled 分支与 impl 权重后处理委托，影响所有 MLA 后端。

```
def process_weights_after_loading(self, act_dtype: torch.dtype):
    # 先委托给具体 impl 做后端特有的权重后处理（默认 no-op），与 Attention 类
    # 的同名回调模式对齐；AMX 后端在这里利用 kv_b_proj 推导并 VNNI 打包
    # W_UK/W_UV，供 prefill/decode 的 bmm_cpu 复用。
    self.impl.process_weights_after_loading(act_dtype)

    if self.is_amx_bmm_enabled:
        # AMXMLAImpl 已把 raw weight 打包成 VNNI 格式，通用路径不再需要额外的
        # bf16 W_UK_T/W_UV 副本，释放原始权重以节省内存。
        self.kv_b_proj.weight = torch.nn.Parameter(
            torch.empty(0), requires_grad=False
        )
        return
    # 非 AMX 路径保持原逻辑：从 raw weight 推导 W_UK_T/W_UV 的 bf16 备份，
    # 并预留量化 bmm 的扩展位（如 is_aiter_triton_fp8_bmm_enabled 分支）。
    ...
```

csrc/cpu/sgl-kernels/mla_cache.cpp

实现 CPU 版 concat_and_cache_mla，将 kv_c_normed 与 k_pe 写入同一 576 宽 latent cache 行，补齐 GPU-only 算子在 CPU 的缺口。

```
// vLLM 自己的 CPU 版 MLA cache 写入 op：GPU 专用 concat_and_cache_mla 在 CPU
// 没有注册，这里仿照 SGLang 的 store_cache_cpu 实现，但推广为把两个源张量
// （kv_c_normed、k_pe）写入同一 cache 行的两个不同列区间 —— MLA 的 cache 是
// 一条 576 宽（512 + 64）的 latent 行，直接复刻 SGLang 的双 tensor 版本不成立。
template <typename scalar_t, typename index_t>
void concat_and_cache_mla_kernel_impl(
    const scalar_t* __restrict__ kv_c_normed, // [num_tokens, kv_lora_rank]
    const scalar_t* __restrict__ k_pe, // [num_tokens, qk_rope_head_dim]
    scalar_t* __restrict__ kv_cache, // [.., kv_lora_rank + qk_rope_head_dim]
    const index_t* __restrict__ slot_mapping, // [num_tokens]
    int64_t num_tokens, int64_t kv_lora_rank, int64_t qk_rope_head_dim,
    int64_t kv_c_stride, int64_t k_pe_stride, int64_t cache_stride) {
  at::parallel_for(0, num_tokens, 0, [&](int64_t begin, int64_t end) {
    for (int64_t i = begin; i < end; ++i) {
      int64_t slot = static_cast<int64_t>(slot_mapping[i]);
      if (slot < 0) {
```

```
        // 负槽位表示 padding token，跳过写入 —— 与 GPU concat_and_cache_mla 语义一致。
        continue;
      }
      scalar_t* __restrict__ cache_row = kv_cache + slot * cache_stride;
      // 先写 kv_c_normed 覆盖前 kv_lora_rank 列，再写 k_pe 覆盖后续 rope 列。
      copy_stub(cache_row, kv_c_normed + i * kv_c_stride, kv_lora_rank);
      copy_stub(cache_row + kv_lora_rank, k_pe + i * k_pe_stride, qk_rope_head_dim);
    }
  });
}
```

评论区精华

本次 review 的 diff 级评论很少：claude[bot] 因 fork 仓库自动跳过，jikunshang 直接 approve。真正的讨论集中在 issue 评论与 commit message 中。maobaolong 在 issue 中感谢作者引入此 PR 提升 CPU MLA 性能，并说明其 PR #49453 无法通过 vllm + dsv2 的 e2e 测试，本 PR 修复该问题，同时请求 review 其关联 PR #51471。提交历史中体现了最重要的设计取舍：fused_qkv_rope 路径因 torch.compile 追踪区域调用 convert_weight_packed（无 fake-tensor/meta kernel）而崩溃，且 DeepSeek-V2-Lite 的 q_lora_rank=None 掩盖了问题，最终决定丢弃 fused 路径；get_mla_prefill_backend 的 is_cpu 检查最初放在最前面，导致大量 stub 了 current_platform 的 GPU 测试误入 CPU 分支，修复为移入 device_capability is None 分支内。

- get_mla_prefill_backend 的 CPU 分支位置修复 (correctness): 把 is_cpu 检查移入已有的 device_capability is None 分支（CpuPlatform 不覆盖 get_device_capability，必然返回 None），保持真实 CPU 行为不变，同时避免 mock 干扰。
- fused_qkv_rope AMX 路径因 torch.compile 崩溃被丢弃 (performance): 删除 fused 路径，避免在 compile 模式下引入自定义 op 融合，改回复用既有 qkv 投影。
- decode absorb 复用 AMX 打包权重并释放原始权重 (performance): AMX 后端 decode/prefill 共用 bmm_cpu 打包权重，并释放 kv_b_proj raw weight 节省内存。
- metadata 计算从逐层提升到每 step 一次 (performance): 所有 per-step 不变的派生元数据在 builder 中算一次并挂在 metadata 上供各层读取。
- 社区确认本 PR 修复 #49453 的 e2e 问题 (other): 无代码变更；确认了共享 bugfix 的价值对社区可见。
- decode 调用中 loc 参数被证明为死参数 (correctness): decode_attention_cpu 的 loc 参数改为可选，AMX 路径不再分配。

风险与影响

- 风险：
  1. 共享核心路径变更：mla_attention.py 的 process_weights_after_loading 现在委托 self.impl，并新增 is_amx_bmm_enabled 分支，所有 MLA 后端都会经过这条委托路径，若其他后端有未预期的自我实现可能产生行为变化（当前默认后端无覆盖，为 no-op）。linear.py 的 _cpu_keep_raw_weight 只影响 CPU 打包路径。

2. 后端选择正确性：cpu.py 依赖 torch.cpu._is_amx_tile_supported() 决定自动切换 AMX_MLA，该 API 在虚拟化 / 容器环境可能误报；AMX_MLA 仅支持 bf16 与 block_size 为 32 的倍数，配置不满足时构造会 raise NotImplementedError。
3. vendored C++ 内核维护成本：decode.cpp（1821 行）、extend.cpp 等来自 SGLang 且有本地针对性修改（如 extend 的因果 mask 修复、mla_cache 的双列偏移写入），上游同步时需小心合并；且仅 AVX512-BF16/VNNI 目标编译，非 AMX CPU 不覆盖。
4. 数值精度与性能：bf16 链路使 backend 级测试容忍度放宽到 1.5e-1；AMX 路径显式断言 kv_cache_dtype == "auto"，未支持 fp8 KV cache；TP 扩展吞吐受跨 socket all-reduce 开销限制，呈现次线性扩展。
5. 测试覆盖盲区：test_amx_mla.py 在非 AMX 环境整体 skip，CI 只有 x86 AMX 机器执行，ARM 和旧 x86 无针对性回归验证（虽有 test_cpu_attn.py 1585 passed 覆盖共享代码对 CPU_ATTN 的影响）。 - 影响：对用户：在带 AMX 的 x86 CPU 上运行 DeepSeek V2/V3/R1 的用户会在不改变 API 的情况下自动获得 AMX_MLA 后端，bench 显示 1000 prompts 输出吞吐 816.9 tok/s、总吞吐 4084.3 tok/s；非 AMX 主机行为不变（回退 CPU_MLA）。对系统：新增约 3000 行 vendored C++ kernel 与 4 个自定义算子，CPU 构建新增 AMX 门控；MLA 公共代码的 bug 修复惠及所有平台，尤其解决 CPU 上 vllm + dsv2 的 e2e 崩溃。对团队：为 CPU 平台建立了与 GPU 对等的高性能 MLA 后端范式，后续可复用该模式扩展 fp8 KV cache、GQA 分块等特性。 - 风险标记：共享 MLA 核心路径变更，仅 AMX 主机收益，vendored C++ 内核维护成本，非 AMX 环境测试静默跳过，torch.compile 兼容性隐患

关联脉络

- PR #49453 [CPU] Add reference CPU MLA backend (CPUMLABackend): PR body 与 issue 评论明确引用：本 PR 是在该参考实现基础上的高性能替代，且修复了 maobaolong 报告的被 #49453 引入的 vllm + dsv2 e2e 崩溃问题。
- PR #51471 [CPU] Fix e2e test for vllm + dsv2: maobaolong 在 issue 评论中请求 review 该 PR，它与本 PR 修复的 kv_b_proj 权重释放及 CPU prefill backend 选择问题直接相关。