PR #49453 完整报告

vllm-project/vllm

[CPU] Add MLA backend so DeepSeek-V2/V3 can run on CPU

合并时间：2026-08-06 16:51

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/49453

执行摘要

- 一句话：CPU 新增 MLA 后端，DeepSeek-V2/V3 可在 CPU 运行
- 推荐动作：值得精读。重点关注三处设计决策：① 如何通过继承 MLACommonBackend/Impl 复用共享 MLA 脚手架，同时用 " 跳过父类 __init__ + 手动复制属性 " 规避 GPU kernel 依赖；② 将 CPU 专属的 KV cache 写入逻辑内联进 do_kv_cache_update 而非污染公共算子；③ ARM 与 x86 在 Vectorized<float> 默认构造语义上的差异如何导致隐性 bug。该 PR 也是后续 CPU MLA 性能优化的起点。

功能与动机

PR body 明确说明：让 DeepSeek-V2/V3 风格 MLA 模型可在 CPU 平台运行，这是一条参考质量路径（正确性优先、性能后置），让没有 GPU 的人也能在本地体验 MLA 模型。

实现拆解

1. 新增 CPUMLA 后端模块（vllm/v1/attention/backends/mla/cpu_mla.py）：CPUMLABackend 描述符声明支持 head_size=576、block_size=16，并注册为 AttentionBackendEnum.CPU_MLA（见 registry.py）。CPUMLAImpl 继承 MLACommonImpl 但刻意绕过父类 __init__（父类会尝试选择 GPU prefill kernel 并在 CPU 上抛错），手动复制父类所需的全部属性；forward_mqa 将 q 拼接为连续张量后调用 ops.mla_decode_kvcache_cpu，do_kv_cache_update 用纯 PyTorch 写入 latent cache 并跳过 -1 padding slot。
2. 平台接线与配置约束（vllm/platforms/cpu.py）：get_attn_backend_cls 在 use_mla 时返回 AttentionBackendEnum.CPU_MLA.get_path() 而非直接抛 NotImplementedError；check_and_update_config 对 MLA 模型强制 block_size=16（即使覆盖用户显式设置也给出 warning）、禁用 chunked prefill 与 prefix caching、拉高 max_num_batched_tokens；pack_kv_cache 对三维 latent cache 直接 early-return，避免误走 key/value 拆分。
3. 修复两个隐藏 CPU bug：csrc/cpu/cpu_types_arm.hpp 中 ARM 的 FP32/BF16 fma 重载原先调用 fmadd 但丢弃返回值（fmadd 不就地修改参数），改为 acc = fmadd(a, b, acc)；csrc/cpu/mla_decode.cpp 中 acc_vec / this_out 显式零初始化，避免在默认构造为 trivial 的 ARM 后端上累加未定义值。
4. 测试配套（tests/v1/attention/test_cpu_mla_backend.py）：test_kv_cache_cpu_write 不依赖权重，校验 latent cache 写入与 -1 padding 槽位处理；test_cpu_mla_backend_smoke 用 hf_overrides 缩小 DeepSeek-V2-Lite 并以 dummy 权

重端到端跑 prefill+decode，标记 cpu_model 使非 CPU CI 跳过。

关键文件：

- vllm/v1/attention/backends/mla/cpu_mla.py（模块 MLA 后端；类别 source；类型 core-logic；符号 CPUMLABackend, CPUMLAImpl, forward_mqa, do_kv_cache_update）：新增的 CPU MLA 后端核心文件，包含 CPUMLABackend 描述符与 CPUMLAImpl 实现，是本次功能的主体。
- tests/v1/attention/test_cpu_mla_backend.py（模块 测试配套；类别 test；类型 test-coverage；符号 test_kv_cache_cpu_write, test_cpu_mla_backend_smoke）：新增测试覆盖两种层级：不依赖权重的 KV cache 写入测试，以及带 hf_overrides 缩小模型的端到端 smoke 测试，是功能正确性的主要保障。
- vllm/platforms/cpu.py（模块 CPU 平台；类别 source；类型 core-logic；符号 get_attn_backend_cls, check_and_update_config, pack_kv_cache）：CPU 平台层负责 MLA 后端的选取、block_size 强制约束、chunked prefill / prefix caching 禁用以及 latent cache 的 pack 跳过，是功能落地的关键接线。
- csrc/cpu/cpu_types_arm.hpp（模块 向量算子；类别 source；类型 core-logic；符号 fma）：修复 ARM 平台 FP32/BF16 FMA 指令返回值被丢弃的 bug，该 bug 导致注意力分数在 ARM 上恒为零，是本 PR 能跑通的关键前提。
- csrc/cpu/mla_decode.cpp（模块 CPU 内核；类别 source；类型 core-logic；符号 mla_decode_block_head）：CPU MLA decode 内核的累加器与输出缓冲显式零初始化，修复 ARM 上默认构造不置零导致的垃圾累加。
- vllm/v1/attention/backends/registry.py（模块 后端注册；类别 source；类型 configuration；符号 AttentionBackendEnum）：注册 CPU_MLA 后端枚举，使平台层能按路径解析到 cpu_mla.py 中的实现。

关键符号：CPUMLABackend.get_supported_head_sizes, CPUMLABackend.get_supported_kernel_block_sizes, CPUMLABackend.supports_block_size, CPUMLAImpl.init, CPUMLAImpl.forward_mqa, CPUMLAImpl.do_kv_cache_update, CpuPlatform.get_attn_backend_cls, CpuPlatform.check_and_update_config, CpuPlatform.pack_kv_cache, vec_op::fma, mla_decode_block_head

关键源码片段

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

新增的 CPU MLA 后端核心文件，包含 CPUMLABackend 描述符与 CPUMLAImpl 实现，是本次功能的主体。

```
def do_kv_cache_update(
    self,
    kv_c_normed: torch.Tensor,
    k_pe: torch.Tensor,
    kv_cache: torch.Tensor,
    slot_mapping: torch.Tensor,
    kv_cache_dtype: str,
```

```
    k_scale: torch.Tensor,
) -> None:
    # CPU 上不需要 fp8 量化，因此 kv_cache_dtype 和 k_scale 不使用。
    # 该纯 PyTorch 路径替代了 CUDA-only 的 concat_and_cache_mla 算子。
    if kv_cache.numel() == 0:
        return
    k_pe_sq = k_pe.squeeze(1)
    kv_lora_rank = kv_c_normed.shape[-1]
    pe_dim = k_pe_sq.shape[-1]
    # latent cache 布局为 [N, block_size, kv_lora_rank + pe_dim]，
    # 先展平以便用 slot 索引写入。
    flat = kv_cache.view(-1, kv_lora_rank + pe_dim)
    slots = slot_mapping.flatten().to(torch.long)
    # 调度器会把 padding 槽位置为 -1，必须跳过，否则负索引会
    # 回绕到缓存末尾并覆盖真实 token 的数据。
    valid_mask = slots >= 0
    if not valid_mask.all().item():
        slots = slots[valid_mask]
        kv_c_normed = kv_c_normed[valid_mask]
        k_pe_sq = k_pe_sq[valid_mask]
    target_dtype = flat.dtype
    # 前半段写 kv_c（压缩后的 latent 表示），后半段写 k_pe（RoPE 部分）。
    flat[slots, :kv_lora_rank] = kv_c_normed.to(target_dtype)
    flat[slots, kv_lora_rank:] = k_pe_sq.view(-1, pe_dim).to(target_dtype)
```

vllm/platforms/cpu.py

CPU 平台层负责 MLA 后端的选取、block_size 强制约束、chunked prefill / prefix caching 禁用以及 latent cache 的 pack 跳过，是功能落地的关键接线。

```
@classmethod
def get_attn_backend_cls(
    cls,
    selected_backend: "AttentionBackendEnum",
    attn_selector_config: "AttentionSelectorConfig",
    num_heads: int | None = None,
) -> str:
    if attn_selector_config.use_sparse:
        raise NotImplementedError("Sparse Attention is not supported on CPU.")
    if attn_selector_config.use_mla:
        # 参考级 CPU MLA 实现：性能不是目标，只是把 decode 内核与 SDPA
        # prefill 接到共享 MLA 脚手架上，让 DeepSeek 风格模型能跑起来。
        if selected_backend and selected_backend != AttentionBackendEnum.CPU_MLA:
            logger.info("Cannot use %s backend on CPU.", selected_backend)
        return AttentionBackendEnum.CPU_MLA.get_path()
    if selected_backend and selected_backend != AttentionBackendEnum.CPU_ATTN:
        logger.info("Cannot use %s backend on CPU.", selected_backend)
    return AttentionBackendEnum.CPU_ATTN.get_path()
```

csrc/cpu/cpu_types_arm.hpp

修复 ARM 平台 FP32/BF16 FMA 指令返回值被丢弃的 bug，该 bug 导致注意力分数在 ARM 上恒为零，是本 PR 能跑通的关键前提。

```
// ARM 与 x86 在 fma 语义上的差异：x86 的 fma 是 `acc = acc + a * b`（赋值），
// 而 at::vec::fmadd(a, b, c) 返回 a * b + c，不修改任何参数。
// 旧代码直接调用 fmadd 但丢弃返回值，导致 FMA 实际是 no-op，
// 最终 attention score 在 ARM 上全部为 0。
inline void fma(FP32Vec16& acc, FP32Vec16& a, FP32Vec16& b) {
  // 必须把返回值写回 acc，否则累加不会发生。
  acc.reg.val[0] = fmadd(a.reg.val[0], b.reg.val[0], acc.reg.val[0]);
  acc.reg.val[1] = fmadd(a.reg.val[1], b.reg.val[1], acc.reg.val[1]);
  acc.reg.val[2] = fmadd(a.reg.val[2], b.reg.val[2], acc.reg.val[2]);
  acc.reg.val[3] = fmadd(a.reg.val[3], b.reg.val[3], acc.reg.val[3]);
}
```

评论区精华

Review 中 bigPYJ1151 提出两个关键意见，作者均已采纳：

- 对 cpu_mla.py 中 _run_prefill_new_tokens 的存在提出疑问（疑似仅存在于该文件的自定义路由），作者回复 "Removed it."，最终删除该函数，prefill 统一走父类公共逻辑。
- 对 vllm/_custom_ops.py 中新增的 CPU fallback 建议 "The part can be directly moved into do_kv_cache_update"，作者回复 "Done."，最终实现将 KV cache 写入逻辑内联进后端，避免在公共算子入口增加平台分支。
- bigPYJ1151 最终 APPROVED，团队评审通过。
- _run_prefill_new_tokens 路由是否正确 (design): 作者回复 "Removed it."，最终删除了该自定义函数，prefill 统一走父类公共路径。
- CPU KV cache fallback 的位置 (design): 作者回复 "Done." 并采纳，KV cache 写入逻辑最终内联进 CPUMLAImpl.do_kv_cache_update。

风险与影响

- 风险：
  - block_size 强制覆盖：check_and_update_config 在 MLA 场景下无条件将 block_size 设为 16，即使覆盖用户显式设置，可能让用户困惑；且当前 decode 内核仅支持该尺寸，后续扩展受限。
  - 父类属性同步风险：CPUMLAImpl.__init__ 手动复制 MLACommonImpl 的属性清单（如 q_lora_rank、kv_b_proj、dcp_world_size 等），若父类未来新增属性，此处可能遗漏，PR body 已明确提示需保持同步。
  - ARM FMA 修复的影响面：cpu_types_arm.hpp 的 fma 是 vec_op 公共算子，修复会影响所有 ARM 平台上使用该算子的路径（如普通 attention），但这是从“静默错误”到“正确结果”的修复，风险低。
  - 性能与功能限制：SDPA prefill 无 chunked prefill / prefix caching，长上下文内存占用高；仅支持 head_dim=576 与 block_size=16，其他 MLA 变体（如更大 head_dim）无法运行。

    - 测试覆盖：端到端 smoke 测试仅在 CPU CI 执行（cpu_model 标记），且使用 dummy 权重，无法验证数值正确性；KV cache 写入测试未覆盖真实模型权重下的类型转换路径。
- 影响：
    - 用户：无 GPU 用户（尤其 macOS ARM 用户）可以本地运行 DeepSeek-V2/V3 做功能验证、教学演示和小模型 smoke 测试，但性能明显弱于 GPU。
    - 系统：CPU 平台配置逻辑发生变更，MLA 模型的 KV cache 布局从 K/V 对切换为单张 latent 张量，pack_kv_cache 等平台工具需要识别三维张量并跳过；非 MLA 的 CPU 模型行为不变。
    - 团队：为后续 CPU MLA 性能优化（融合 prefill、放宽 block_size/head_dim 限制）奠定基础；同时修复的 ARM FMA bug 惠及所有 ARM CPU 用户，属于附带收益。
    - 风险标记：CPU 平台核心路径变更，block_size 强制覆盖用户配置，父类属性手动复制易失步，ARM 数值行为修复影响面广，性能非目标且受限

关联脉络

- PR #50578 [ROCm][MLA] Use asm decode for non-divisor small head counts: 同为 MLA 后端在非 NVIDIA 平台（ROCm）上的适配，与本 PR 的 CPU 适配属于同一功能演进线，可对照平台特化思路。
- PR #50613 [Attention][MLA] Per-request scheduling for MLA chunked context: MLA chunked prefill 调度优化，与本 PR 在 CPU 上禁用 chunked prefill 形成对比，展示不同平台对 MLA 预填充策略的取舍。