PR #48791 完整报告

vllm-project/vllm

[ModelRunner V2] Enable sequence pooling for embedding and classification models

合并时间：2026-07-29 21:30

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/48791

执行摘要

- 一句话：MRV2 支持序列级 embed/classify 池化执行
- 推荐动作：值得精读。这是 MRV2 pooling 迁移的核心执行 PR，其中三个设计决策值得学习：一是用子类隔离 encoder-only 注意力并复用 AttentionGroup 机制，二是用 CPU 侧 finished_mask 避免 GPU->CPU 往返，三是测试以 Hugging Face 为独立参考而不是 V1 互相对照。建议结合 #49331、#48290 连起来看，掌握 MRV2 启用流程的完整链路。

功能与动机

Issue #41286 的 MRV2 迁移路线图中，pooling 模型是最后一段未落地的领域；PR body 明确指出主分支 V1 能初始化分类模型，但强制 MRV2 会触发 PoolingRunner 第 26 行的断言 assert "embed" in model.pooler.get_supported_tasks()，且分类打分 V2_SCORE 与 V1_SCORE 从 8.308 漂移到 -7.424。本 PR 就是要修复这两个 gap：让 MRV2 真正执行序列级 embed/classify pooler，并把打分拉回与 Hugging Face 一致的数值。

实现拆解

1. PoolingRunner 从“硬编码 LAST”改为“执行模型 pooler”（vllm/v1/worker/gpu/pool/pooling_runner.py）：构造函数接收 VllmConfig，通过 model_config.get_pooling_task() 选定任务，并限定在 _SUPPORTED_TASKS = {"embed", "classify"} 内，token 级任务（token_embed/token_classify）在初始化时抛出带 VLLM_USE_V2_MODEL_RUNNER=0 提示的 ValueError；新增 add_request()/remove_request() 维护 per-request 的 PoolingParams、PoolingStates 与 prompt_token_ids（仅 requires_token_ids 时保存，控制内存占用）；pool() 调用 model.pooler(hidden_states, pooling_metadata)，借助 build_pooling_cursor() 支持 chunked prefill，并返回 CPU 侧 finished_mask；dummy_pooler_run() 改为逐任务构造 dummy PoolingMetadata 并捕获 OOM 给出可操作提示，覆盖 warmup/profile 真实 pooler 路径。
2. EncoderOnlyModelState 支持 paired-input token_type_ids（vllm/v1/worker/gpu/model_states/encoder_only.py）：add_request() 从 pooling_params.extra_kwargs["compressed_token_type_ids"] 解析 query/document 配对边界，生成 int32 的 token_type_ids 序列并按 req_id 保存，remove_request() 同步清理；prepare_inputs() 在 chunked prefill 下按 num_computed_tokens 偏移切片填充 token_type_ids，pinned memory 非阻塞拷贝到 GPU，使 cross-encoder 分类打分在

MRV2 下与 V1/HF 一致。

3. 异步输出支持异构 PoolerOutput（vllm/v1/worker/gpu/async_utils.py）：AsyncPoolingOutput 构造参数从 is_valid: torch.Tensor | None 改为 finished_mask: list[bool]，避免 GPU->CPU 回拷 valid mask；当输出为单个张量且全部请求完成时走整张量 to("cpu", non_blocking=True) 快路径，否则逐请求 unbind() 并过滤未完成项，支持 Matryoshka 等 768/256 维度不同的异构输出。
4. ModelRunner 与 warmup 集成（vllm/v1/worker/gpu/model_runner.py、warmup.py）：load_model() 传入 vllm_config 构造 PoolingRunner；add_requests()/_remove_request() 桥接 pooling_runner 的请求生命周期；pool() 使用新的 finished_mask；warmup 按 get_pooling_task() 选中的实际任务构造 PoolingParams，替代固定 PoolingParams()，并删除不可达的 pooling_task is None 分支。
5. 测试配套：新增 forced-MRV2 与 Hugging Face/SentenceTransformers 的 parity 测试（BERT 分类 / 配对打分 test_classification.py、encoder MEAN 与 Matryoshka 混合维度 test_embedding.py、chunked prefill 序列 embed test_all_pooling_plus_chunked_prefill.py）；PoolingRunner 元数据单测（test_splade_sparse_pooler.py）覆盖 required token_ids 收集、按需存储与不支持任务拒绝；streaming 测试补 pooling_runner = None mock。

关键文件：

- vllm/v1/worker/gpu/pool/pooling_runner.py（模块 池化执行；类别 source；类型 core-logic；符号 PoolingRunner.init, PoolingRunner.add_request, PoolingRunner.remove_request, PoolingRunner._get_pooling_metadata）：核心执行路径：从硬编码 last-token 归一化改为调用模型 pooler，新增 per-request 元数据、chunked prefill 游标与按任务 warmup，是本次变更的主体。
- vllm/v1/worker/gpu/model_states/encoder_only.py（模块 编码器状态；类别 source；类型 data-contract；符号 EncoderOnlyModelState.add_request, EncoderOnlyModelState.remove_request, EncoderOnlyModelState.prepare_inputs）：为 encoder-only 模型补齐 paired-input 的 token_type_ids 传递，是分类配对打分与 V1/HF 对齐的关键数据契约。
- vllm/v1/worker/gpu/async_utils.py（模块 异步输出；类别 source；类型 core-logic；符号 AsyncPoolingOutput.init, AsyncPoolingOutput.get_output）：AsyncPoolingOutput 改造为支持异构 PoolerOutput，并基于 CPU finished_mask 处理未完成请求，避免 GPU->CPU 回拷。
- vllm/v1/worker/gpu/model_runner.py（模块 模型运行器；类别 source；类型 dependency-wiring；符号 ModelRunner.load_model, ModelRunner.add_requests, ModelRunner._remove_request, ModelRunner.pool）：PoolingRunner 构造、请求生命周期桥接与 pool() 返回值适配，是把新执行路径接入 ModelRunner 的枢纽。
- tests/models/language/pooling/test_embedding.py（模块 嵌入测试；类别 test；类型 test-coverage；符号 test_encoder_model_runner_v2, test_matryoshka_dimensions_model_runner_v2）：新增 encoder MEAN embedding 与 Matryoshka 混合维度（768/256）的 forced-MRV2 对 HF/SentenceTransformers parity 测试。

- tests/models/language/pooling/test_splade_sparse_pooler.py（模块 池化测试；类别 test；类型 test-coverage；符号 test_pooling_runner_gathers_required_token_ids, test_pooling_runner_stores_only_required_token_ids, test_pooling_runner_rejects_unsupported_selected_task）：为 PoolingRunner 元数据逻辑补充单元测试：required token_ids 收集、按需存储、不支持任务拒绝。

关键符号：PoolingRunner.init, PoolingRunner.add_request, PoolingRunner.remove_request, PoolingRunner._get_pooling_metadata, PoolingRunner.pool, PoolingRunner._dummy_pooler_run_task, EncoderOnlyModelState.add_request, EncoderOnlyModelState.remove_request, EncoderOnlyModelState.prepare_inputs, AsyncPoolingOutput.init, AsyncPoolingOutput.get_output

关键源码片段

vllm/v1/worker/gpu/model_states/encoder_only.py

为 encoder-only 模型补齐 paired-input 的 token_type_ids 传递，是分类配对打分与 V1/HF 对齐的关键数据契约。

```
# EncoderOnlyModelState 在 DefaultModelState 基础上扩展 paired-input 支持。
# token_type_ids 标记 query/document 分界，供 cross-encoder 分类打分使用。
class EncoderOnlyModelState(DefaultModelState):
    # __init__ 中新增 self.token_type_ids: dict[str, torch.Tensor] = {}

    def add_request(self, req_index: int, new_req_data: NewRequestData) -> None:
        super().add_request(req_index, new_req_data)
        pooling_params = new_req_data.pooling_params
        if pooling_params is None or pooling_params.extra_kwargs is None:
            return

        # 解析前端传入的压缩配对边界（compressed_token_type_ids 为 document 起始下标），
        # 生成 int32 的 token_type_ids 序列并与 req_id 绑定。
        token_type_start = pooling_params.extra_kwargs.get("compressed_token_type_ids")
        if token_type_start is not None:
            assert new_req_data.prompt_token_ids is not None
            self.token_type_ids[new_req_data.req_id] = (
                torch.arange(len(new_req_data.prompt_token_ids), dtype=torch.int32)
                >= token_type_start
            ).to(torch.int32)

    def remove_request(self, req_id: str) -> None:
        super().remove_request(req_id)
        self.token_type_ids.pop(req_id, None)

    def prepare_inputs(
        self, input_batch: InputBatch, req_states: RequestState
    ) -> dict[str, torch.Tensor | None]:
        model_inputs = super().prepare_inputs(input_batch, req_states)
```

```
    if not self.token_type_ids:
        return model_inputs

    # chunked prefill 下按 num_computed_tokens 偏移切片，逐请求填充 token_type_ids，
    # pinned memory + 非阻塞拷贝，避免阻塞主计算流。
    token_type_ids_cpu = torch.zeros(
        input_batch.num_tokens_after_padding,
        dtype=torch.int32,
        pin_memory=PIN_MEMORY,
    )
    offset = 0
    for i, req_id in enumerate(input_batch.req_ids):
        num_tokens = int(input_batch.num_scheduled_tokens[i])
        request_token_type_ids = self.token_type_ids.get(req_id)
        if request_token_type_ids is not None:
            start = int(input_batch.num_computed_tokens_np[i])
            token_type_ids_cpu[offset : offset + num_tokens].copy_(
                request_token_type_ids[start : start + num_tokens]
            )
        offset += num_tokens
    model_inputs["token_type_ids"] = token_type_ids_cpu.to(
        self.device, non_blocking=True
    )
    return model_inputs
```

评论区精华

核心讨论集中在设计隔离与异步路径：njhill 建议将 encoder-only 逻辑隔离为子类并单独开 PR（#49331），本 PR 直接以其为基础，避免改动 DefaultModelState 与非 pooling 路径；同时建议避免 GPU->CPU 回拷 valid mask，改为 CPU 侧确定 finished_mask，作者在 async_utils.py 中落实。LucasWilkinson 指出 prompt_token_ids_cpu 应使用 pinned memory 才能保证真正异步复制，作者改为 PIN_MEMORY；并建议测试以 HF 为独立参考而不是 V1 对照（因 MRV1 即将弃用），作者将 classification/embedding 测试改为 forced MRV2 vs HF。yewentao256 建议测试文件不用 cast 改用 MagicMock、内联单次使用的常量，njhill 建议删除 _dummy_pooler_run 返回值与 warmup 中不可达分支，均已采纳。

- prompt_token_ids_cpu 应使用 pinned memory (performance): 作者确认修复，在 _get_pooling_metadata() 中改用 pin_memory=PIN_MEMORY，并补充了 pin 状态断言测试。
- 测试参考对象：V1 对照 vs Hugging Face (testing): 作者将分类与嵌入测试改为 forced MRV2 对 HF/SentenceTransformers 独立参照，并保留 V1 仅用于回归基线。
- 异步输出避免 GPU->CPU 回拷 valid mask (design): 实现为 AsyncPoolingOutput 接收 finished_mask: list[bool]，去掉 is_valid 张量回拷逻辑。
- encoder-only 逻辑隔离为子类 (design): 本 PR 基于 #49331 的 EncoderOnlyModelState 子类实现，避免改动 DefaultModelState 与公共路径。
- 测试文件避免 cast 类型断言 (style): 作者改用 MagicMock 构造测试对象，移除 cast 依赖。

- _dummy_pooler_run 返回值与 warmup dead code (style): 作者删除返回值与不可达分支，简化 warmup 逻辑。

风险与影响

- 风险： 1) 内存与异步复制：PoolingRunner 为需 token_ids 的请求在 CPU 保存完整 prompt_token_ids 并构造 padded 张量，max_prompt_len 取 batch 内最大值，长尾请求会放大内存；CPU 侧已用 PIN_MEMORY，但需确认所有 to(device, non_blocking=True) 路径都持有对应的 CPU 引用（AsyncPoolingOutput 已保留）。 2) chunked prefill 偏移：encoder_only.py 的 prepare_inputs() 依赖 num_computed_tokens_np 偏移切片 token_type_ids，若调度器计算语义变化可能越界，已有 chunked 测试覆盖 25/27 token 用例，但更大 chunk 数仍需观察。 3) 异构输出分支：async_utils.py 中 isinstance(pooler_output, torch.Tensor) 与逐项 unbind 的二分逻辑，若有新 pooler 返回混合类型可能漏处理，当前 Matryoshka 测试只覆盖了 Tensor 分支。 4) 兼容性边界：token_embed/token_classify 在强制 MRV2 下直接报错，属于预期行为，但需要 #48290 默认启用时保证 V1 回退路径清晰。
  - 影响：对用户：BERT/RoBERTa 等 encoder 模型的 embedding、分类与重排打分现在可在 MRV2 下获得与 Hugging Face 一致的数值，且 NFCorpus 重排精度 0.3288 与 V1 及 SentenceTransformers 0.32898 几乎一致；Matryoshka 混合维度输出、chunked prefill 场景均得到覆盖。对系统：PoolingRunner 从占位实现变为真实执行路径，warmup 覆盖所有支持任务，为后续默认启用 MRV2 扫清障碍；对团队：这是 #41286 路线图的重要里程碑，后续 #48290 只需切换默认开关，token 级任务仍由 follow-up PR 承接。
  - 风险标记：核心路径变更，异步复制正确性，异构输出分支，兼容性边界

关联脉络

- PR #49331 [ModelRunner V2] Support encoder-only attention: 本 PR 的直接基础：njhill 在讨论中提出并单独落地的 encoder-only attention 子类，本 PR 在其之上实现 pooler 执行。
- PR #48290 [ModelRunner V2] Enable MRV2 for pooling models by default: 本 PR 的 PR body 明确说明修复 #41286 的 sequence-level 部分并解锁 #48290；#48290 是默认启用开关，本 PR 是前置执行能力。
- PR #52425 [ModelRunner v2] Support Transformers pooling model: 同属 MRV2 pooling 功能线，后续扩展 Transformers 后端 pooling 模型路由，与本 PR 的 PoolingRunner 能力互补。